PAIRING-BASED CRYPTOSYSTEMS AND KEY AGREEMENT PROTOCOLS

By

Zhaohui Cheng

SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
MIDDLESEX UNIVERSITY
LONDON, UNITED KINGDOM
MARCH, 2007

Table of Contents

List of Tables

List of Figures

Abstract

For a long time, pairings on elliptic curves have been considered to be destructive in elliptic curve cryptography. Only recently after some pioneering works, particularly the well-known Boneh-Franklin identity-based encryption (IBE), pairings have quickly become an important tool to construct novel cryptographic schemes.

In this thesis, several new cryptographic schemes with pairings are proposed, which are both efficient and secure with respect to a properly defined security model, and some relevant previous schemes are revisited.

IBE provides a public key encryption mechanism where a public key can be an arbitrary string such as an entity identifier and unwieldy certificates are unnecessary. Based on the Sakai-Kasahara key construction, an IBE scheme which is secure in the Boneh-Franklin IBE model is constructed, and two identity-based key encapsulation mechanisms are proposed. These schemes achieve the best efficiency among the existing schemes to date.

Recently Al-Riyami and Paterson introduced the certificateless public key encryption (CL-PKE) paradigm, which eliminates the need of certificates and at the same time retains the desirable properties of IBE without the key escrow problem. The security formulation of CL-PKE is revisited and a strong security model for this type of mechanism is defined. Following a heuristic approach, three efficient CL-PKE schemes which are secure in the defined strong security model are proposed.

Identity-based two-party key agreement protocols from pairings are also investigated. The Bellare-Rogaway key agreement model is enhanced and within the model several previously unproven protocols in the literature are formally analysed. In considering that the user identity may be sensitive information in many environments, an identity-based key agreement protocol with unilateral identity privacy is proposed.

Acknowledgements

I would like to thank Richard Comley, my supervisor, for his many suggestions and constant support during this research. I also want to thank Luminita Vasiu and Orhan Gemikonakli for their direction in this study.

Liqun Chen shared with me her knowledge of cryptography and provided much useful advice and friendly encouragement in the last three years.

During the programme, I had fruitful discussions with many colleagues; among them are Alex Dent, John Malone-Lee, Chris Mitchell, Nigel Smart, Qiang Tang and many others.

I also want to thank Chris Kindberg and Richard Read for proofreading some chapters of this thesis.

Finally I am grateful to my family for their patience and love. Without their support this work would never have come into existence.

Zhaohui Cheng

Chapter 1

Introduction

In the literature, many cryptographic schemes such as [69, 72] were constructed on finite Abelian groups. As the points on the elliptic curve form an Abelian group, it is natural to consider using elliptic curves in cryptography. In the mid-1980s, Victor Miller [120] and Neal Koblitz [105] independently proposed elliptic curve cryptography (ECC). Point groups of elliptic curves exhibit some particular properties which make them more attractive than other groups. For example, the subexponential-time index calculus algorithm of the discrete logarithm problem (DLP) [157] does not work on the elliptic curve point groups. The known algorithms for the general elliptic curve discrete logarithm problem (ECDLP), such as the Baby-Step-Giant-Step algorithm and the Pollard's Rho algorithm [127], are running in exponential-time. Hence for those cryptographic schemes based on the hardness of DLP, they can be implemented on elliptic curve point groups with smaller size than the multiplicative group of a finite field. This results in better performance of certain schemes and smaller bandwidth cost as well. Gradually, ECC gathered strength in practice partially due to this advantage.

However, not all the elliptic curves have such advantage. In 1991 Menezes, Okamoto and Vanstone [126] found that for certain curves called supersingular curves, by using the Weil pairing [140], ECDLP can be converted to DLP over some extension of the ground field on which the curve is defined, and so can be solved by the faster subexponential-time

algorithm. This algorithm is named as the MOV reduction. Later, it was shown that the MOV reduction can be applied to ordinary curves as well when the embedding degree (see the definition in Section 2.5.2) is sufficiently small [155]. Hence, in practice one is advised to avoid using curves with small embedding degrees when implementing ECCs[1].

However, recent developments on the pairing-based cryptography created another "twist". Although the MOV reduction implies faster algorithms to solve the ECDLP when the embedding degree is small, certain security levels for practical use can still be achieved if the ground field size and the embedding degree are chosen properly. Moreover, in this context efficient pairings can be applied in very constructive ways to build novel cryptographic schemes. For example, Joux [102] observed that the pairing can be used to construct a round-efficient tripartite key agreement protocol. By using the pairing computation and a Diffie-Hellman-like message exchange, the Joux protocol requires each party to broadcast only a single message to establish an agreed session key among three parties. An even more striking development is that in 2001 Boneh and Franklin [19] using pairing constructed the first practical identity-based encryption (IBE) scheme with provable security based on a reasonable pairing-based intractability assumption. This work has solved the long-standing cryptographic problem of constructing a secure and practical IBE as proposed by Shamir in 1984 [138]. After these pioneering works, many novel cryptographic systems have been constructed with pairings. This thesis is devoted to the study of pairing-based cryptography.

In the last three decades, we have learned an important lesson that *ad hoc* approaches to construct cryptographic schemes are a dangerous way to go. Numerous security schemes with only *ad hoc* security analysis were later broken. As an alternative, many researchers have been seeking to build cryptography on firm foundations by borrowing methods from the theory of complexity. Rigorous approaches to treat cryptography have then prevailed. Now it has become more or less a standard process to solve a cryptographic problem with

[1]It was also shown that for abnormal curves, there exist polynomial-time algorithms for the DLP [142, 148, 141] and the Weil descent attack is applicable to certain normal curves [83].

two phases: a *definitional* phase and a *constructive* phase [77]. In the definitional phase, one identifies the functionality of a cryptographic task and rigorously defines the cryptographic problem to capture natural security concerns. Then, in the constructive phase, one designs a cryptographic scheme to satisfy the definition with a rigorous proof based on a computational complexity assumption. There are two categories of constructive activities. One is to demonstrate that a cryptography definition is achievable in principle by constructing schemes based on some fundamental assumptions such as the existence of one-way functions. Those constructions may be general but often inefficient. The other aims to construct schemes suitable for practical applications. The constructions of the second category make use of stronger but reasonable assumptions and in return deliver attractive performance for the practitioner. Over the years, there have been many cryptographic schemes constructed in the literature, but only few have made it from theory into practice [107]. To increase the likelihood that results are applied in practice, this thesis focuses on cryptographic schemes from pairings which not only meet robust security definitions based on reasonable intractability assumptions but also achieve high efficiency.

In this work, three cryptographic primitives from pairings are studied, namely, the identity-based encryption, the certificateless public key encryption, and the identity-based key agreement protocol.

To simplify the management of public keys, in 1984 Shamir proposed the concept of identity-based cryptography (IBC) in which the identity of an entity serves as the public key of the entity directly; therefore unwieldy certificates are unnecessary and a public key infrastructure (PKI) to issue certificates is not required. The construction of secure and practical IBE had been an open problem for years. Finally, in 2001 Boneh and Franklin constructed the first practical and security-provable IBE using pairings. After that, a number of IBEs have been proposed, e.g. [10, 12, 162, 82]. Though significant improvement has been made on speeding up the pairing computation [24, 22], pairing is still a very heavy

operation and pairing-based schemes are an order of magnitude slower than the traditional elliptic curve cryptography. Since the performance seriously affects a cryptography scheme's application, particularly in environments with devices of limited power, it is important to design IBE schemes which are efficient as possible. We present another IBE which is secure in terms of the Boneh-Franklin IBE security definition and is highly efficient. Based on the basic version of the IBE scheme, we further propose two identity-based key encapsulation mechanisms (ID-KEM), which working with a data encapsulation mechanism (DEM) can encrypt arbitrarily long messages securely. These schemes are the fastest among all the published schemes from pairings to date.

The IBC following Shamir's formulation inherently exhibits the key escrow property. In many environments this property is undesirable. In 2003 Al-Riyami and Paterson introduced the certificateless public key cryptography (CL-PKC) paradigm [5]. CL-PKC is an intermediate model between traditional certificated PKC and IBC. It eliminates the need of certificates and at the same time retains the desirable properties of IBC without the key escrow problem. We revisit the Al-Riyami-Paterson security definition of certificateless public key encryption (CL-PKE) and define a strong security model for this type of encryption. There have been a number of constructions of CL-PKE in the literature, some of which are not very efficient and others are insecure. By making a simple observation on some existing IBEs and PKEs, we propose a heuristic approach to constructing efficient CL-PKEs. Following this approach, using three IBEs, we construct three concrete CL-PKEs, which are secure in the defined strong model and are the most efficient one derived from the three types of IBE respectively. We also present a general conversion which can transform a weak CL-PKE to a scheme satisfying the strong definition.

The key agreement protocol, as a fundamental cryptographic primitive, allows parties to establish secret keys over a network controlled by adversaries. Since the publication of Sakai *et al.*'s non-interactive identity-based key agreement (IB-KAP) from pairings [152],

many identity-based two-party key agreement protocols using pairings have been proposed. However, some elegant and efficient schemes are either not formally analysed, such as the Smart-Chen-Kudla protocol [63] and the Shim-Yuan-Li protocol [166], or the security reductions are seriously flawed such as the McCallugh-Barreto protocols [124, 125]. As demonstrated by numerous cases in the literature, the key agreements designed in *ad hoc* modes are often vulnerable to certain attacks. Formal security analysis is necessary for one to build confidence in the security of those schemes, particularly considering that the Shim-Yuan-Li protocol is derived from a protocol suffering from serious attacks. We first enhance the Bellare-Rogaway key agreement security model to cover all the commonly required security properties. Then we prove the security of those three schemes in the defined strong model. In considering that the user identity may be a piece of sensitive information in certain environments, we construct a new identity-based protocol from pairings with unilateral identity privacy. This protocol, apart from establishing an authenticated secret, can hide a user's identity from an adversary.

1.1 Outline of the Thesis

The thesis consists of three major parts dealing with three topics: efficient IBEs, efficient CL-PKEs and efficient IB-KAPs respectively.

In Chapter 2, before presenting those pairing-based cryptographic protocols, we introduce some preliminaries which are relevant for the subsequent chapters. We begin with introducing the concept of public key cryptography by focusing on techniques of authentic public key distribution. Then we explain the provable security paradigm, the random oracle model and our view on a security reduction in the random oracle model. After that we define pairing and present the pairing instances on elliptic curves which serves as the basic tool for the cryptographic schemes throughout the thesis. Finally we present the pairing-related computational complexity assumptions which we will use to prove the security of

the schemes presented in the thesis.

In Chapter 3, based on the Sakai-Kasahara's key construction [151], we first construct a complete identity-based encryption SK-IBE, which is secure in terms of the Boneh-Franklin IBE security definition. We then extend the basic version of the scheme to two ID-KEMs. Through a detailed performance comparison, we demonstrate that these schemes achieve the best efficiency among existing IBEs. We finally investigate the impact of Cheon's algorithm to solve the strong Diffie-Hellman problem [41] on the proposed schemes.

In Chapter 4, we revisit the Al-Riyami-Paterson security definition of CL-PKE and define a strong security model for this paradigm. Then by making a simple observation on some existing IBEs and PKEs, we propose a heuristic approach to constructing efficient CL-PKEs from IBEs. Following this approach, we construct three efficient concrete CL-PKEs which are secure in the defined security model.

In Chapter 5, we first enhance the Bellare-Rogaway key agreement model to cover all of the common security properties. Then we formally analyse a few existing protocols including the enhanced Smart protocol, the enhanced Shim protocol and the McCallugh-Barreto protocol in the enhanced security model. Finally we construct an IB-KAP with unilateral identity privacy which is suitable for those environments where identity privacy is important.

1.2 Previous Publications

The efficient IBE scheme of Section 3.4 originally appeared in "Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme", joint work with Liqun Chen [48] presented at the 10th IMA International Conference of Cryptography and Coding 2005. The efficient ID-KEM scheme of Section 3.5.3 originally appeared in "An Efficient ID-KEM Based On The Sakai–Kasahara Key Construction", joint work with Liqun Chen, John Malone-Lee and Nigel Smart, which was published in IEE Proceedings Information Security [51].

The efficient CL-PKE schemes of Section 4.5 originally appeared in "Efficient Certificateless Public Key Encryption", joint work with Richard Comley, which was posted on IACR ePrint in January 2005 [46].

The security analysis of the enhanced Smart protocol and the enhanced Shim protocol of Section 5.4 originally appeared in "Identity-based Key Agreement Protocols from Pairings", joint work with Liqun Chen and Nigel Smart [52] to appear in International Journal of Information Security. The security analysis of the McCullagh-Barreto protocols [124, 125] of Section 5.5 originally appeared in "On Security Proof of McCullagh-Barreto's Key Agreement Protocol and its Variants", joint work with Liqun Chen to appear in International Journal of Security and Networks - Special Issue on Cryptography in Networks [47]. The protocol with unilateral identity privacy of Section 5.6 originally appeared in "Identity-Based Key Agreement with Unilateral Identity Privacy Using Pairings", joint work with Liqun Chen, Richard Comley and Qiang Tang [50] presented at ISPEC 2006.

Chapter 2

Preliminaries

In this chapter the necessary preliminaries are described. First a brief recap of the public key cryptography and the public key authenticity problem in the public key cryptography is presented. Then the provable security paradigm and our view on a security proof is explained. After that pairing, which serves as the basic tools for the cryptographic schemes throughout the thesis, is formally defined and two pairing instances on elliptic curves are presented. Finally the pairing-related computational complexity assumptions which are used to prove the security of the schemes in the thesis are defined.

2.1 Public Key Cryptography

After Diffie and Hellman pointed out the new direction of cryptography in 1976 [69], the public key cryptography has become one fundamental stream of cryptography and has been widely used in practice to provide information security.

In contrast with symmetric-key cryptography, an entity in the public-key cryptosystem has two keys: One is publicly-known and the other is kept secret to the entity itself. The entity uses the private key and others use the public key in the cryptographic operations. For example, for encryption, the others use the entity's public key to encrypt messages, while the entity uses its private key to decrypt the ciphertexts. A public-key encryption can be formally specified by three algorithms:

- **Key Generation** $\mathbb{G}_{\mathtt{PKE}}(1^k)$: On input 1^k, which is a formal notation for a machine with running time polynomial in k, the probabilistic algorithm generates the public/private key pair.

$$(K_{pub}, K_{priv}) \leftarrow \mathbb{G}_{\mathtt{PKE}}(1^k),$$

  where K_{pub} is the public key and K_{priv} is the associated private key.

- **Encrypt** $\mathbb{E}_{\mathtt{PKE}}(K_{pub}, m;\ r)$: The algorithm encrypts a message m from the message space $\mathbb{M}_{\mathtt{PKE}}(K_{pub})$.

$$C \leftarrow \mathbb{E}_{\mathtt{PKE}}(K_{pub}, m;\ r),$$

  where C is the ciphertext in the ciphertext space $\mathbb{C}_{\mathtt{PKE}}(K_{pub})$. This probabilistic algorithm uses the randomness r from the random space $\mathbb{R}_{\mathtt{PKE}}(K_{pub})$.

- **Decrypt** $\mathbb{D}_{\mathtt{PKE}}(K_{pub}, K_{priv}, C)$: The deterministic algorithm decrypts a ciphertext and outputs value of the message m or a failure symbol $\perp$.

$$(m \text{ or } \perp) \leftarrow \mathbb{D}_{\mathtt{PKE}}(K_{pub}, K_{priv}, C).$$

Various security notions have been defined for a PKE scheme. The commonly accepted definitions are based on one of the two-stage games in Table 2.1 between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of the encryption algorithm and a challenger.

In the games, s is some state information and $\mathcal{O}_{\mathtt{PKE}}$ are oracles to which the adversary has access. There are various possibilities for these oracles depending on the attack model:

- Chosen Plaintext Attack (CPA) Model: In this model the adversary has no oracle access.

- Adaptive Chosen Ciphertext Attack (CCA2) Model: In this model the adversary has access to a decryption oracle which on query C returns $\mathbb{D}_{\mathtt{PKE}}(K_{pub}, K_{priv}, C)$. There is one restriction on how the adversary uses this oracle: In the second phase $\mathcal{A}_2$ is not allowed to call the decryption oracle with C^*.

Table 2.1: PKE Security Formulation

OW Adversarial Game	IND Adversarial Game
1. $(K_{pub}, K_{priv}) \leftarrow \mathbb{G}_{\mathtt{PKE}}(1^k)$.	1. $(K_{pub}, K_{priv}) \leftarrow \mathbb{G}_{\mathtt{PKE}}(1^k)$.
2. $s \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathtt{PKE}}}(K_{pub})$.	2. $(s, m_0, m_1) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathtt{PKE}}}(K_{pub})$.
3. $m \leftarrow \mathbb{M}_{\mathtt{PKE}}(K_{pub}), r \leftarrow \mathbb{R}_{\mathtt{PKE}}(K_{pub})$.	3. $b \leftarrow \{0,1\}, r \leftarrow \mathbb{R}_{\mathtt{PKE}}(K_{pub})$.
4. $C^* \leftarrow \mathbb{E}_{\mathtt{PKE}}(K_{pub}, m;\ r)$.	4. $C^* \leftarrow \mathbb{E}_{\mathtt{PKE}}(K_{pub}, m_b;\ r)$.
5. $m' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathtt{PKE}}}(K_{pub}, C^*, s)$.	5. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathtt{PKE}}}(K_{pub}, C^*, s, m_0, m_1)$.

Let MOD denote the mode of attack, either CPA or CCA2. The adversary's advantage in the first game is defined to be

$$\mathtt{Adv}_{\mathtt{PKE}}^{\mathtt{OW-MOD}}(\mathcal{A}) = \Pr[m' = m],$$

while, the advantage in the second game is given by

$$\mathtt{Adv}_{\mathtt{PKE}}^{\mathtt{IND-MOD}}(\mathcal{A}) = \mid 2\Pr[b' = b] - 1 \mid .$$

A PKE algorithm is considered to be secure, in the sense of a given goal and attack model (IND-CCA2, for example) if, for all PPT adversaries, the advantage in the relevant game is a negligible function of the security parameter k.

Definition 2.1.1. (**Negligible Function**) A function $\epsilon(k) : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if for every constant $c \geq 0$ there exists a positive integer k' such that $\epsilon(k) < k^{-c}$ for all $k > k'$.

In public key cryptography two fundamental problems should be solved. The first problem is regarding designing secure cryptographic primitives such as encryption, signature or authenticated key exchange to provide information confidentiality, non-repudiation or authentication. The second problem is regarding distributing public keys with authenticity. Without the authenticity of public keys, a public key cryptographic system is vulnerable to an impersonation attack. For example the impersonation attack on a PKE system works as

follows: The adversary *Eve* generates a pair of keys (K'_{pub}, K'_{priv}) and replaces entity *Alice*'s public key K_{pub} with K'_{pub} by certain means such as publishing the information on a white board. When entity *Bob* uses the false public key K'_{pub} to encrypt messages to *Alice*, *Eve* can certainly decrypt the ciphertexts with the knowledge of K'_{priv}. Hence, *Bob* has to know *Alice*'s real public key before encrypting any message to *Alice*.

There have been several solutions to the above second problem. The common strategy is to introduce a third party (an authority) which is trusted by both *Alice* and *Bob*. The authority authenticates an entity's ownership of a public key by some means. Based on the role the authority can play and the information the authority knows, Girault defined three security levels [76].

At level 1, the authority knows every entity's private key and hence can impersonate any entity without being detected. This is the paradigm where the IBC settles in. In the IBC, each entity uses its identity as the public key (hence the public key authenticity problem is trivial), while the private key is generated by an algorithm from the identifier and a piece of secret information maintained by the authority. Hence the IBC is a natural key-escrow system.

At level 2, the authority does not know an entity's real private key but can still impersonate the entity without being detected. The CL-PKC introduced by Al-Riyami and Paterson [5] is one of such systems. In the CL-PKC, the private key of an entity is determined by two pieces of secret information from the authority and the entity respectively. Moreover one secret is not computable from the other. Hence the authority does not know an entity's private key. In the CL-PKC the public key generated by an entity is not authenticated by others, hence one can replace others' public keys without being detected. However, without the secret passed from the authority to the legitimate entity, the impersonator still cannot decrypt the ciphertext even with one piece of secret. On the other hand, the authority can still impersonate an entity by generating a false public key and

hence with both pieces of secret information can decrypt a ciphertext generated under the false public key.

At level 3, the authority does not know an entity's private key and if it generates a false warranty that proves one entity owns a specific public key, such behavior is detectable. The cryptographic schemes based on the PKI obtains the level 3 security. The authority called a certificate authority (CA) issues a certificate, which by using a digital signature to prevent certificate forgery, securely binds an entity's identifier with the entity's public key. In the PKI system, *Bob* obtains and verifies *Alice*'s certificate, so to get *Alice*'s authentic public key before encrypting messages to *Alice*. Because of the unforgeability of the used signature scheme in the certificate generation, the CA will be held to account if a certificate binding an entity with a false public key is issued.

Arguably, the level 3 security is the most attractive one in most cases. However, the PKI system faces various challenges in practical deployment [80]. The system complexity has been the major obstacle to its wider application. On the other hand, in certain circumstances, either level 1 or 2 security is acceptable, and maybe the level 1 system (with the key escrow property) is even mandatory. Moreover, the IBC and CL-PKC offer much more simplified systems. Hence it has been of great interest to design such types of secure system with practical efficiency. In Chapter 3, 4 and 5 a number of efficient IBEs, CL-PKEs and identity-based authenticated key exchange protocols are presented respectively.

2.2 Provable Security

Since the publication of the Diffie-Hellman work, many public-key cryptographic schemes have been designed, with more or less heuristic proofs of their security relative to some computational complexity assumptions. However, these proposed schemes were often broken, sometimes years after they were first presented.

Arguably starting from the Goldwasser-Micali's landmark work [85], a different paradigm,

provable security, was introduced into modern cryptography. In this paradigm, a cryptographic scheme is analysed by providing a proof in the framework of complexity theory. Specifically, the proof provides a reduction from a well-studied hard problem, such as the RSA or the discrete logarithm problem, to an adversary against a cryptographic scheme in a properly defined security model. A valid proof vindicates the security of a protocol in the way that if in the defined security model there exists a polynomial-time adversary against the scheme, there will be a polynomial-time algorithm to solve the hard problem on which the proof is based; and turn the logic around: If the underlying problem is hard, then the scheme is secure regarding the security model. Hence such proofs are also called security reductions.

In the first few years, the work of this line was of great theoretical importance. Researchers defined various security notions by formulating the corresponding security models and then designed schemes to achieve these notions. Schemes were designed and analysed in such ways that only very fundamental assumptions, such as the existence of one-way function were made. Those works answered the fundamental questions that whether certain security notions can be achieved in the framework of computational complexity theory. On the other hand, those cryptographic schemes with provably strong security based on common intractability assumptions are often not practically efficient.

In the early 1990's, Bellare and Rogaway formally introduced another instrument "random oracle model" [31] which simulates ideal hash functions. In the random oracle model, to design a scheme, one assumes that all parties (including the adversary) share an ideal random function. The ideal random function is interpreted as a program g with an initially empty vector v and a sub-routine t. Program g on query input x and the auxiliary input y works as follows:

```
Input: v, x, y
Ouput: g(v,x,y)
    If x is in the vector v, i.e. v[x] is not empty,
    then
        return v[x];
    else
        begin
            v[x]=t(v, x, y);
            return v[x];
        end
```

where `t(v,x,y)` generates a truly random value on inputs v, x and y. To construct a real cryptographic scheme one replaces the random oracle with a "good" cryptographic hash function, i.e. providing all parties (including the adversary) with the description of this function.

The random oracle model has far-reaching influence on the later cryptographic scheme design. Many efficient schemes were designed and their security was proven in the random oracle model, such as RSA-OAEP [33, 75].

As to the soundness of this model, no one has ever provided a convincing contradiction to its practical validity. So far, only a few theoretical counter-examples [54, 130, 13] have been presented. These counter-examples are either with easy-pick flaws in designs, or just to meet artificial security notions purposely. There has been no scheme with valid proof in the random oracle model reported being broken in practice except some with improper implementations [122, 103]. Hence, the model has been largely accepted by the community as a good instrument to validate the security of a scheme.

Following the line of provable security, in this thesis the security of the presented schemes will be formally analysed, and all the analysis will be conducted in the random oracle model.

2.3 Two Views on Security Reductions

There are two views on a security proof in the framework of computational complexity: asymptotic and concrete view. An asymptotic view on a security reduction is that the

reduction just guarantees the security of a system in general, but does not suggest a security parameter at any concrete security level. On the contrary, the concrete view is that a security analysis not only vindicates the security of a scheme, but also allows one based on the quantitative result of a reduction to estimate the value of the security parameter for which the scheme is secure at a chosen security level.

In this thesis we adopt the asymptotic view on a reduction in the random oracle model[1] for the following reasons. First, the random oracle model essentially is just a heuristic instrument to help security analysis. This has been demonstrated in [54]. The algorithm constructed in a reduction with random oracles cannot serve as a hard problem solver even if an adversary algorithm against the scheme is constructible in practice. Second, a reduction with random oracle queries is almost unlikely to be tight if without resorting to random oracle's distinctive properties. These properties include the programmability, which allows one to choose certain parameters adaptively to optimise the reduction performance, e.g. Theorem 4.1 in [19], the true randomness and the access to the random oracle query/response list, which allows the potential stronger gap assumptions for better tightness in theory. However, all these properties are not preserved by a cryptographic hash function which replaces a random oracle in a concrete implementation. On the one hand, a loose security reduction often cannot suggest any meaningful security parameter for a practical implementation. On the other hand, some tricks played with the random oracle resulting in a tight reduction seem not to be generally accepted, see the interesting comment in Section 4.3 [111] on Katz-Wang's "magic bit" trick [113] played in a signature for a tight reduction in the random oracle model. A similar trick was played by [4] resulting in an IBE with a tight reduction.

Indeed, if a reduction result is related to hash queries, the quantitative result could be meaningless for choosing a security parameter with an efficient implementation in practice.

[1] As the asymptotic view is adopted, for the simplicity of analysis some negligible quantity may be ignored in some reductions in Chapter 5 of the thesis.

For example, the well-known BF-IBE has the following simplified security result [79]: Assume the adversary $\mathcal{A}$ queries each random oracle q_H times and wins the the adaptive chosen ciphertext attack game defined in [19] (see Section 3.2 for details) with advantage $\epsilon(k)$, then there exists an algorithm $\mathcal{B}$ to solve the Bilinear DH (BDH) problem (see Definition 2.6.4 in Section 2.6) with the following advantage and time:

$$\mathtt{Adv}_{\mathcal{B}}^{\mathtt{BDH}}(k) \approx \frac{\epsilon(k)}{q_H^2}$$
$$t_{\mathcal{B}} \approx t_{\mathcal{A}} + 640 q_H t_H$$

where t_H is the cost of the hash function.

Notice that the BasicPub in [19] is IND-CPA secure instead of merely OW-CPA secure. By this fact, we get a tighter reduction than the one in [79]. For simplicity we further assume the BDH problem has the same computational complexity as the DL problem and all the operations including the hash query, decryption, encryption, extraction and group multiplication use equivalent unit time.

It is generally believed that in an attack 2^{50} hash queries should be considered possible (i.e. $4q_H \approx 2^{50}$ in BF-IBE) and 2^{30} other queries such as decrypt or sign should be presumed. For the reduction to be meaningful, i.e. the reduction implies an algorithm with better performance than the general DL algorithm $\mathcal{D}$, which can recover the master secret key so as to totally break the scheme, it should have $\mathtt{Adv}_{\mathcal{B}}^{\mathtt{BDH}}(k) \geq \mathtt{Adv}_{\mathcal{D}}^{\mathtt{BDH}}(k)$, when both $\mathcal{B}$ and $\mathcal{D}$ run in the time of $t_{\mathcal{B}} \approx 2^{57}$ operations. It turns out that

$$\frac{\epsilon(k)}{2^{96}} \geq \frac{2^{57}}{2^k}$$

We assume $\epsilon(k) = 1/2^{10}$, i.e. an adversary with 2^{50} operations can break the scheme with probability $1/2^{10}$ by the adaptive chosen ciphertext attacks. The security parameter $k = 80$ is generally considered to be the proper choice for this security level. However, the above reduction result implies that the BF-IBE should be implemented at least with $k = 163$. Such an implementation is far from efficient in practice for the proposed security

level. Even if we assume that a hash operation is thirty-two thousand times faster than a group multiplication, k is still around 128, which implies that by using 128-bit security level parameters in theory, the reduction can only guarantee the 80-bit security level.

2.4 Pairing

In this section, the pairing which is the basic tool to construct the cryptographic schemes in the thesis is formally defined. First some notations that will be used throughout the thesis are listed.

- $\mathbb{G}_1$: an additive cyclic group of prime order p.
- $\mathbb{G}_2$: an additive group of exponent p, whose order is some power of p.
- $\mathbb{G}_t$: a multiplicative cyclic group of prime order p.
- ψ: a homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.
- P_2: an element of order p in $\mathbb{G}_2$
- P_1: a generator of $\mathbb{G}_1$ and $\psi(P_2) = P_1$.
- $\mathbb{Z}_p$: the modulo group with elements from $\{0, 1, \ldots, p-1\}$ and $\mathbb{Z}_p^*$ with elements from $\{1, \ldots, p-1\}$

Definition 2.4.1. A **pairing** is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$ on three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$, which has the following properties:

1. Bilinearity: for all $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$ and for all $(a, b) \in \mathbb{Z}_p \times \mathbb{Z}_p$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Non-degeneracy: there exist non-trivial points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$ both of order p such that $\hat{e}(P, Q) \neq 1$.
3. Computability: for all $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$, $\hat{e}(P, Q)$ is efficiently computable.

The groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$, the elements P_1 and P_2, the pairing $\hat{e}$, and possibly the homomorphism ψ are referred as a set of pairing parameters. The subscript of an element is used to refer to the the group where the element is chosen.

2.5 Pairings on Elliptic Curve

Here we define the elliptic curve and introduce the basic facts of the point group and the pairings on elliptic curves.

2.5.1 Elliptic Curve

Let K be a field and $\overline{K}$ be the algebraic closure of K. Let $a_i \in K$ and E be an elliptic curve defined by

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

over K, that is

$$E = \{(x, y) \in \overline{K}^2 \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, a_i \in K\} \cup \{\mathcal{O}\},$$

where $\mathcal{O}$ is a special point, called *the point at infinity.*

Let L be an algebraic extension of K. Define

$$E(L) = \{(x, y) \in L^2 \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, a_i \in K\} \cup \{\mathcal{O}\}.$$

We say that the point (x, y) in $E(L)$ is L-rational.

The addition operation of two points $U, V \in E$ is defined as follows:

1. Draw a straight line though U and V to find the third intersecting point W on the curve,

2. Draw a vertical line through W to find the intersecting point Z on the curve,

and define $U + V = Z$. If $U = V \neq \mathcal{O}$, then the straight line in Step 1 is the tangent line of the curve through U. When U adding $\mathcal{O}$, the straight line in Step 1 is the vertical line passing through U, and the vertical line in Step 2 is same as the line in Step 1, which intersects the curve at the same point U. So, $U + \mathcal{O} = U$.

By the above addition rules, the addition of points on an elliptic curve E has following properties:

- Associativity: $P_1 + (P_2 + P_3) = (P_1 + P_2) + P_3$ for all $P_i \in E$.
- Identity: $P + \mathcal{O} = P$ for all points $P \in E$.
- Inverse: for all $P \in E$, there exists $-P$ with $P + (-P) = \mathcal{O}$.
- Commutativity: $P_1 + P_2 = P_2 + P_1$ for all $P_i \in E$.

So the points on the elliptic curve E form an additive Abelian group with $\mathcal{O}$ as identity, and $E(L)$ is the subgroup of E.

All the points in $E(L)$ whose order is a divisor of n, i.e. all $P \in E(L)$ with $nP = \mathcal{O}$, form a group, called an n-torsion group, denoted by $E(L)[n]$ which is a subgroup of $E(L)$.

2.5.2 Weil Pairing and Tate Pairing

Now we introduce two pairing instances on elliptic curves: the Weil pairing and the Tate pairing.

In the practice of cryptography, we normally set $K = \mathbb{F}_q$ where q is a prime power and implement cryptographic schemes in the group $E(\mathbb{F}_q)[r]$ where r is a prime dividing $\#E(\mathbb{F}_q)$ and $char(\mathbb{F}_q) \nmid r$ ($\#E(\mathbb{F}_q)$ denotes the number of points in $E(\mathbb{F}_q)$ and $char(K)$ is the characteristic of field K). Define $\mu_r = \{x \in \overline{\mathbb{F}}_q \mid x^r = 1\}$, i.e. the group of r-th roots of unity in $\overline{\mathbb{F}}_q$. Let k be the smallest positive integer such that $r \mid q^k - 1$. k is called the embedding degree of $E(\mathbb{F}_q)[r]$ and $\mu_r \subseteq \mathbb{F}_{q^k}^{*}$[2]. $E(\mathbb{F}_{q^k})[r]$ is a group of exponent r and is isomorphic to $\mathbb{Z}_r \times \mathbb{Z}_r$. Define $rE(\mathbb{F}_{q^k}) = \{rP \mid P \in E(\mathbb{F}_{q^k})\}$ and then the quotient group $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$ is also of exponent r and isomorphic to $\mathbb{Z}_r \times \mathbb{Z}_r$.

Definition 2.5.1. The **Weil pairing** is defined as a map

$$e_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})[r] \rightarrow \mu_r.$$

The Weil pairing satisfies the following properties:

[2]Hitt [94] noted that μ_r is also embedded in $\mathbb{F}_{t^x}$ when $q = t^m$ for some prime t and x is the least positive integer such that $r|t^x - 1$. This may affect the security of schemes with certain pairing parameters if $t^x \ll q^k$.

1. Bilinearity: $e_r(P+R,Q) = e_r(P,Q) \cdot e_r(R,Q)$ and $e_r(P,Q+S) = e_r(P,Q) \cdot e_r(P,S)$ for all $P, Q, R, S \in E(\mathbb{F}_{q^k})[r]$.

2. Non-degeneracy: if $e_r(P,Q) = 1$ for all $P \in E(\mathbb{F}_{q^k})[r]$ then $Q = \mathcal{O}$ and also that if $e_r(P,Q) = 1$ for all $Q \in E(\mathbb{F}_{q^k})[r]$ then $P = \mathcal{O}$.

3. $e_r(P,P) = 1$ for all $P \in E(\mathbb{F}_{q^k})[r]$.

4. $e_r(P,Q) = e_r(Q,P)^{-1}$ for all $P, Q \in E(\mathbb{F}_{q^k})[r]$.

Definition 2.5.2. Suppose $E(\mathbb{F}_{q^k})[r]$ has a point of order r. The **Tate pairing** is defined as a map

$$\langle \cdot, \cdot \rangle_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \rightarrow \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$$

and the **modified Tate pairing** is defined as a map

$$\tau_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \rightarrow \mu_r.$$

The Tate pairing and the modified Tate pairing has the relation such that $\tau_r(P,Q) = \langle P,Q \rangle_r^{\frac{q^k-1}{r}}$. The modified Tate pairing satisfies the following properties.

1. Bilinearity: $\tau_r(P+R,Q) = \tau_r(P,Q) \cdot \tau_r(R,Q)$ and $\tau_r(P,Q+S) = \tau_r(P,Q) \cdot \tau_r(P,S)$ for all $P, R \in E(F_{q^k})[r]$ and $Q, S \in E(F_{q^k})/rE(F_{q^k})$.

2. Non-degeneracy: for each $P \neq \mathcal{O} \in E(F_{q^k})[r]$, there exists $Q \in E(F_{q^k})/rE(F_{q^k})$ such that $\tau_r(P,Q) \neq 1$.

As the modified Tate pairing τ_r has unique result for each pair of inputs and this is generally required by cryptographic schemes, in the sequel we use the Tate pairing to refer to τ_r to be used in the schemes presented.

Both the Weil and Tate pairing are polynomial-time computable and great efforts have been made to speed up these pairing computation [84, 24, 121, 22, 98]. The Tate pairing is relatively more efficient regarding the chosen parameters in practice [87]. Some other pairings such the Eta [22] and Ate [98] pairings could perform even faster than the Tate pairing in some circumstances.

2.5.3 Four Types of Pairing

When the Weil or the Tate pairing is used, the choice of $\mathbb{G}_1, \mathbb{G}_2$ is flexible and different choices may significantly affect a cryptographic scheme's implementation and performance.

Hence it is useful to further specify different types of pairing system by considering different choices of $\mathbb{G}_1$ and $\mathbb{G}_2$. We will consider four types of pairing, three of which are taken from [88] whilst the fourth is from [139].

Let $\mathcal{G} = E[p]$ which includes the points of prime order p on the elliptic curve E defined over $\mathbb{F}_q$ and $\mathcal{O}$. We assume that the group $E[p]$ is contained in $E(\mathbb{F}_{q^k})$, where for simplicity (and efficiency) we further assume that k is even. The group $\mathcal{G}$ is a product of two cyclic groups $\mathcal{G}_1$, $\mathcal{G}_2$ of order p. We can take a point $\mathcal{P}_1 \in E(\mathbb{F}_q)[p]$ as a generator of $\mathcal{G}_1$ and a point $\mathcal{P}_2 \in E(\mathbb{F}_{q^k})$ as a generator of $\mathcal{G}_2$.

The representation of $\mathcal{G}_2$ may also affect the performance of pairings on ordinary curves. A standard method is to represent a point in $\mathcal{G}_2$ as a point on a twist curve of E, and if necessary map the representation point into $\mathcal{G}_2$. For example, the points on the quadratic twist of E over $\mathbb{F}_{q^{k/2}}$ can be mapped onto $E(\mathbb{F}_{q^k})$ using a standard homomorphism. We select $\mathcal{P}_2$ so that it is an image mapped from a point on the quadratic twist of E over $\mathbb{F}_{q^{k/2}}$.

The following is a concrete example from [156]. Suppose an elliptic curve E is defined over $\mathbb{F}_q$ for a prime $q > 3$ as follows:

$$y^2 = x^3 - 3x + b,$$

where $b \in \mathbb{F}_q$. Let $k = 2d$ and let χ denote an element of $\mathbb{F}_{q^d}$ which is quadratic non-residue. The quadratic twist of E over the field $\mathbb{F}_{q^d}$ is defined as E':

$$\chi y^2 = x^3 - 3x + b.$$

Then we can define an injective homomorphism as follows:

$$\Phi : \begin{cases} E'(\mathbb{F}_{q^d}) & \longrightarrow & E(\mathbb{F}_{q^k}) \\ (x, y) & \longmapsto & (x, \sqrt{\chi}y). \end{cases}$$

It is not difficult to see that $E'(\mathbb{F}_{q^d})$ contains a subgroup of order p. Let this subgroup be generated by a point $P' \in E'(\mathbb{F}_{q^d})$. Then we set $\mathcal{P}_2 = \Phi(P')$. Two groups $\langle \mathcal{P}_1 \rangle$ and

$\langle \Phi(P') \rangle$ are linear independent, i.e. any non-identity element in $\langle \mathcal{P}_1 \rangle$ is not a multiple of an element in $\langle \Phi(P') \rangle$. Smart and Vercauteren [156] showed that $\langle \mathcal{P}_1 \rangle$ and $\langle \Phi(P') \rangle$ are the only two subgroups of $E(\mathbb{F}_{q^k})$ with order p that consist of elements with x-coordinate contained in $\mathbb{F}_{q^d}$. We note that inputs with x-coordinate contained in $\mathbb{F}_{q^d}$ can speed up pairing computation, and other representations of $\mathcal{G}_2$ on a twist of higher degree can perform even better [28, 98].

Let the Frobenius automorphism ϕ be defined as follows:

$$\phi : \begin{cases} E(\overline{\mathbb{F}}_q) & \longrightarrow & E(\overline{\mathbb{F}}_q) \\ (x, y) & \longmapsto & (x^q, y^q), \\ \mathcal{O} & \longmapsto & \mathcal{O}. \end{cases}$$

The trace map

$$\mathrm{Tr} : \begin{cases} E(\mathbb{F}_{q^k}) & \longrightarrow & E(\mathbb{F}_q) \\ P & \longmapsto & \sum_{i=1}^{k} \phi^i(P), \end{cases}$$

defines a group homomorphism on $E[p]$ which has kernel $\mathcal{G}_2$, i.e. the set of points which map to $\mathcal{O}$ under the above map is precisely the group $\mathcal{G}_2$. For all parameter choices of cryptographic interest which we know of the trace map can be very efficiently implemented using only k elliptic curve point additions and a few finite field operations.

Now we are ready to specify pairings $\hat{e}$ satisfying Definition 2.4.1. We set $\mathbb{G}_t = \mu_p$ but will choose $\mathbb{G}_1$ and $\mathbb{G}_2$ from $\mathcal{G}$ differently. We focus on the following four cases.

Definition 2.5.3 (Type 1). In this situation, which corresponds to pairings over supersingular elliptic curves, we can define a pairing using so-called distortion maps [159, 24], by taking $\mathbb{G}_1 = \mathbb{G}_2 = \mathcal{G}_1$. We let $P_1 = P_2 = \mathcal{P}_1$. There is an efficient algorithm to cryptographically hash arbitrary bit strings into $\mathbb{G}_1$ and $\mathbb{G}_2$ and (a trivial) group isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ mapping P_2 to P_1.

The following three types correspond to pairings over ordinary elliptic curves.

Definition 2.5.4 (Type 2). In this situation, we take $\mathbb{G}_1 = \mathcal{G}_1$ and $\mathbb{G}_2$ to be a subgroup of $\mathcal{G}$ which is not equal to either $\mathcal{G}_1$ or $\mathcal{G}_2$. We set $P_1 = \mathcal{P}_1$ and for convenience we set $P_2 = \frac{1}{k}\mathcal{P}_1 + \mathcal{P}_2$. There is an efficient algorithm to cryptographically hash arbitrary bit strings into $\mathbb{G}_1$, but there is no way to hash bit strings into $\mathbb{G}_2$ (nor to generate random

elements of $\mathbb{G}_2$ bar multiplying P_2 by an integer). However, there is an efficiently computable group isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ mapping P_2 to P_1, which is simply the trace map restricted to $\mathbb{G}_2$.

Definition 2.5.5 (Type 3)**.** In this situation, we take $\mathbb{G}_1 = \mathcal{G}_1$ and $\mathbb{G}_2 = \mathcal{G}_2$, with generators $P_1 = \mathcal{P}_1$ and $P_2 = \mathcal{P}_2$. There is an efficient algorithm to cryptographically hash arbitrary bit strings into $\mathbb{G}_1$, and a slightly less efficient algorithm to hash bit strings into $\mathbb{G}_2$. However, there is no known efficiently computable group isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ mapping P_2 to P_1.

Definition 2.5.6 (Type 4)**.** In this situation, we take $\mathbb{G}_1 = \mathcal{G}_1$, but we select $\mathbb{G}_2$ to be the whole group $\mathcal{G}$ which is a group of order p^2. As in the Type 2 situation, we set $P_1 = \mathcal{P}_1$ and $P_2 = \frac{1}{k}\mathcal{P}_1 + \mathcal{P}_2$. There is an efficiently computable homomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ such that $\psi(P_2) = P_1$. Hashing into $\mathbb{G}_1$ or $\mathbb{G}_2$ can be performed, although maybe not very efficiently into $\mathbb{G}_2$. However, one cannot hash efficiently into the subgroup of $\mathbb{G}_2$ generated by P_2. Note, that the pairing of a non-zero element in $\mathbb{G}_1$ and a non-zero element in $\mathbb{G}_2$ may be trivial in this situation.

To summarise, in all situations we have that P_1 is the generator of $\mathbb{G}_1$ and P_2 is a fixed element of $\mathbb{G}_2$ of prime order p, such that when there is a computable homomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, we have $\psi(P_2) = P_1$. In Type 3 curves, such an isomorphism does exist but one is just unable to compute it efficiently. We will still refer to ψ in this situation but it should be born in mind that one is unable to compute it efficiently.

2.6 Pairing-Related Complexity Assumptions

Below we list a number of assumptions, some of which have been widely used as the complexity assumptions to construct cryptographic schemes. Each problem is assumed to be defined for a given set of pairing parameters.

Assumption 2.6.1. ***(Diffie-Hellman (DH))*** *For $a, b \in_R \mathbb{Z}_p$ ($\in_R$ denotes randomly sampling) and some values of $i, j, k \in \{1, 2\}$, given (aP_i, bP_j), computing abP_k is hard.*

Assumption 2.6.2. ***(ℓ-Diffie-Hellman Inversion (ℓ-DHI))*** *For a positive integer ℓ, and $\alpha \in_R \mathbb{Z}_p^*$, given $(\alpha P_i, \alpha^2 P_i, \ldots, \alpha^\ell P_i)$, computing $1/\alpha P_i$ is hard.*

Theorem 2.6.1. *[Mitsunari et al. [128]] DH and 1-DHI are polynomial time equivalent, i.e. if there exists a polynomial time algorithm to solve DH, then there exists a polynomial time algorithm for 1-DHI, and if there exists a polynomial time algorithm to solve 1-DHI, then there exists a polynomial time algorithm for DH.*

Assumption 2.6.3. *(ℓ-**Strong Diffie-Hellman (ℓ-SDH)**) For a positive integer ℓ, and $\alpha \in_R \mathbb{Z}_p^*$, given $(\alpha P_i, \alpha^2 P_i, \ldots, \alpha^\ell P_i)$, computing $(h, \frac{1}{\alpha+h} P_i)$ for some $h \in \mathbb{Z}_p^*$ is hard.*

It is easy to show that the the ℓ-DHI problem is easier than the DH problem and ℓ-SDH is considered to be a weaker problem than ℓ-DHI. To gain the confidence on these assumptions, Boneh and Boyen [11] proved a lower bound on the computational complexity of the ℓ-SDH problem in the generic group model [145]: By assuming that the best DL algorithm on the chosen groups is the general DL algorithm, at least $\sqrt{p/\ell}$ group operations are required to solve the ℓ-SDH problem when $\ell < o(\sqrt[3]{p})$.

Assumption 2.6.4. *(**Bilinear Diffie-Hellman (BDH)**) For $a, b, c \in_R \mathbb{Z}_p$, given (aP_i, bP_j, cP_k), for some values of $i, j, k \in \{1, 2\}$, computing $\hat{e}(P_1, P_2)^{abc}$ is hard.*

It is trivial to show that the $\text{BDH}_{i,j,k}$ assumption implies the Diffie-Hellman assumption $\text{DH}_{i,j,k'}$ when $k \neq k'$. Verheul provided evidence that it is likely that the BDH problem is strictly easier than the DH problem on the elliptic curves [159, 110].

There are some other variants of the BDH assumption have been used in the literature.

Assumption 2.6.5. *(**Bilinear Inverse DH (BIDH)**) For $a, b \in_R \mathbb{Z}_p$, given (aP_i, bP_j) for some values of $i, j \in \{1, 2\}$, computing $\hat{e}(P_1, P_2)^{b/a}$ is hard.*

Assumption 2.6.6. *(ℓ-**BDHI**) For a positive integer ℓ, and $\alpha \in_R \mathbb{Z}_p^*$, given $(\alpha P_i, \alpha^2 P_i, \ldots, \alpha^\ell P_i)$ for some value $i \in \{1, 2\}$, computing $\hat{e}(P_1, P_2)^{1/\alpha}$ is hard.*

Theorem 2.6.2. *1-BDHI_2, $\text{BIDH}_{2,2}$ and $\text{BDH}_{2,2,2}$ are polynomial-time equivalent.*

Proof: First, we prove that if there is a polynomial time algorithm $\mathcal{A}$ to solve the 1-BDHI_2 problem, we can construct a polynomial time algorithm $\mathcal{B}$ to solve the $\text{BDH}_{2,2,2}$ problem. Given an instance of $\text{BDH}_{2,2,2}$ problem $(P_1, P_2, aP_2, bP_2, cP_2)$, $\mathcal{B}$ works as follows to compute $\hat{e}(P_1, P_2)^{abc}$.

1. (a) Set $d = 1/(a + b + c)$, which $\mathcal{B}$ does not know.
   (b) Set $Q_2 = (a + b + c)P_2$, $Q_1 = \psi(Q_2)$ and $dQ_2 = P_2$.
   (c) Pass $\mathcal{A}$ the 1-BDHI_2 challenge (Q_1, Q_2, dQ_2), and get
   $$T_1 = \hat{e}(Q_1, Q_2)^{1/d} = \hat{e}(P_1, P_2)^{(a+b+c)^3}.$$
2. Follow item 1 (a) - (c) to get $T_2 = \hat{e}(P_1, P_2)^{a^3}$, $T_3 = \hat{e}(P_1, P_2)^{b^3}$, $T_4 = \hat{e}(P_1, P_2)^{c^3}$, $T_5 = \hat{e}(P_1, P_2)^{(a+b)^3}$, $T_6 = \hat{e}(P_1, P_2)^{(a+c)^3}$ and $T_7 = \hat{e}(P_1, P_2)^{(b+c)^3}$.

3. Compute $\hat{e}(P_1, P_2)^{abc} = (\frac{T_1 \cdot T_2 \cdot T_3 \cdot T_4}{T_5 \cdot T_6 \cdot T_7})^{1/6}$.

Second, we prove that if there is a polynomial time algorithm $\mathcal{A}$ to solve the $\text{BDH}_{2,j,k}$ problem, we can construct a polynomial time algorithm $\mathcal{B}$ to solve the $\text{BIDH}_{2,j}$ problem for $j, k \in \{1, 2\}$. Given an instance of $\text{BIDH}_{2,j}$ problem (P_1, P_2, aP_2, bP_j), $\mathcal{B}$ works as follows to compute $\hat{e}(P_1, P_2)^{b/a}$.

1. Set $a' = 1/a, b' = b/a, c' = 1/a$, which $\mathcal{B}$ does not know.
2. Set $Q_2 = aP_2$, $Q_1 = \psi(Q_2)$.
3. Pass $\mathcal{A}$ the $\text{BDH}_{2,j,k}$ challenge $(Q_1, Q_2, a'Q_2 = P_2, b'Q_j = bP_j, c'Q_2 = P_2)$ if $k = 2$ or $(Q_1, Q_2, a'Q_2 = P_2, b'Q_j = bP_j, c'Q_1 = P_1)$ if $k = 1$, and get $\hat{e}(Q_1, Q_2)^{a'b'c'} = \hat{e}(P_1, P_2)^{b/a}$.

Third, it is easy to show if there is a polynomial time algorithm $\mathcal{A}$ to solve the $\text{BIDH}_{i,j}$ problem, then a polynomial time algorithm $\mathcal{B}$ to solve the 1-BDHI_i can be constructed as follows: Given the 1-BDHI_i challenge (P_1, P_2, aP_i), pass the problem (P_1, P_2, aP_i, P_1) if $j = 1$ or (P_1, P_2, aP_i, P_2) if $j = 2$ to get $\hat{e}(P_1, P_2)^{1/a}$.
This completes the proof. □

The following is another assumption closely related to ℓ-BDHI.

Assumption 2.6.7. ***(Bilinear Collision Attack Assumption (ℓ-BCAA1))*** *For a positive integer ℓ, and $\alpha \in_R \mathbb{Z}_p^*$, given $(\alpha P_i, h_0, (h_1, \frac{1}{h_1+\alpha}P_j), \ldots, (h_\ell, \frac{1}{h_\ell+\alpha}P_j))$ for some values of $i, j \in \{1, 2\}$ where $h_i \in_R \mathbb{Z}_p^*$ and different from each other for $0 \leq i \leq \ell$, computing $\hat{e}(P_1, P_2)^{1/(\alpha+h_0)}$ is hard.*

Theorem 2.6.3. *If there exists a polynomial time algorithm to solve (ℓ-1)-$BDHI_2$, then there exists a polynomial time algorithm for ℓ-$BCAA1_{i,2}$. If there exists a polynomial time algorithm to solve (ℓ-1)-$BCAA1_{i,2}$, then there exists a polynomial time algorithm for ℓ-$BDHI_2$.*

Proof: If there is a polynomial time algorithm $\mathcal{A}$ to solve the (ℓ-1)-BDHI_2 problem, we can construct a polynomial time algorithm $\mathcal{B}$ to solve the ℓ-$\text{BCAA1}_{i,2}$ problem as follows.

Given an instance of the ℓ-$\text{BCAA1}_{i,2}$ problem

$$(\mathbb{G}_1, \mathbb{G}_2, \psi, Q_1, Q_2, yQ_i, h_0, (h_1, \frac{1}{h_1 + y}Q_2), \ldots, (h_\ell, \frac{1}{h_\ell + y}Q_2)),$$

$\mathcal{B}$ works as follows to compute $\hat{e}(Q_1, Q_2)^{1/(y+h_0)}$.

1. Set $x = y + h_0$ which $\mathcal{B}$ does not know, and

$$P_2 = \frac{1}{(y + h_1) \cdots (y + h_\ell)}Q_2.$$

2. For $j = 0, \ldots, \ell - 1$, $\mathcal{B}$ computes

$$x^j P_2 = \frac{(y + h_0)^j}{(y + h_1) \cdots (y + h_\ell)} Q_2 = \sum_{i=1}^{\ell} \frac{c_{ij}}{y + h_i} Q_2$$

where $c_{ij} \in \mathbb{Z}_p$ are computable from h_i's.

3. Set $P_1 = \psi(P_2)$.

4. Pass $\mathcal{A}$ the (ℓ-1)-BDHI$_2$ challenge,

$$(\mathbb{G}_1, \mathbb{G}_2, \psi, P_1, P_2, xP_2, \cdots, x^{\ell-1} P_2),$$

and get $T = \hat{e}(P_1, P_2)^{1/x}$.

5. Set

$$f(z) = \prod_{i=1}^{\ell} (z + h_i - h_0) = \sum_{i=0}^{\ell} d_i z^i$$

where d_i is computable from h_i's and $d_0 \neq 0$ because h_i are different.

6. Note that

$$Q_2 = f(x) P_2 = \sum_{i=0}^{\ell} d_i x^i P_2$$

and

$$\frac{1}{x} Q_2 = \frac{f(x)}{x} P_2 = \sum_{i=0}^{\ell} d_i x^{i-1} P_2$$

.

7. Compute

$$\begin{aligned}
\hat{e}(Q_1, Q_2)^{1/(y+h_0)} &= \hat{e}(\frac{1}{x}\psi(Q_2), Q_2) \\
&= \hat{e}(\sum_{i=0}^{\ell} d_i x^{i-1} \psi(P_2), Q_2) \\
&= T^{d_0^2} \cdot \hat{e}(d_0 P_1, \sum_{i=1}^{\ell} d_i x^{i-1} P_2) \cdot \hat{e}(\sum_{i=1}^{\ell} d_i \psi(x^{i-1} P_2), Q_2).
\end{aligned}$$

If there is a polynomial time algorithm $\mathcal{A}$ to solve the (ℓ-1)-BCAA1$_{i,2}$ problem, we can construct a polynomial time algorithm $\mathcal{B}$ to solve the ℓ-BDHI$_2$ problem as follows.

Given an instance of the ℓ-BDHI$_2$ problem

$$(\mathbb{G}_1, \mathbb{G}_2, \psi, P_1, P_2, xP_2, x^2 P_2, \ldots, x^\ell P_2),$$

$\mathcal{B}$ works as follows to compute $\hat{e}(P_1, P_2)^{1/x}$.

1. Randomly choose different $h_0, \ldots, h_{\ell-1} \in \mathbb{Z}_p^*$ and set $y = x - h_0$ which $\mathcal{B}$ does not know.

2. Let $f(z)$ be the polynomial

$$f(z) = \prod_{i=1}^{\ell-1}(z + h_i - h_0) = \sum_{i=0}^{\ell-1} c_i z^i.$$

The constant term c_0 is non-zero because h_i's are different and c_i is computable from h_i's.

3. Compute

$$Q_2 = \sum_{i=0}^{\ell-1} c_i x^i P_2 = f(x)P_2$$

and

$$yQ_2 = \sum_{i=0}^{\ell-1} c_i x^{i+1} P_2 - h_0 Q_2 = xf(x)P_2 - h_0 Q_2.$$

4. Compute

$$f_i(z) = \frac{f(z)}{z + h_i - h_0} = \sum_{j=0}^{\ell-2} d_j z^j$$

and

$$\frac{1}{y + h_i} Q_2 = \frac{1}{x + h_i - h_0} f(x)P_2 = f_i(x)P_2 = \sum_{j=0}^{\ell-2} d_j x^j P_2$$

for $1 \leq i \leq \ell - 1$.

5. Set $Q_1 = \psi(Q_2)$.

6. Pass the following instance of the $(\ell$-1)-BCAA1$_{i,2}$ problem to $\mathcal{A}$

$$(\mathbb{G}_1, \mathbb{G}_2, \psi, Q_1, Q_2, \psi(yQ_2), h_0, (h_1, \frac{1}{y + h_1} Q_2), \ldots, (h_{\ell-1}, \frac{1}{y + h_{\ell-1}} Q_2)) \text{ if } i = 1$$

or

$$(\mathbb{G}_1, \mathbb{G}_2, \psi, Q_1, Q_2, yQ_2, h_0, (h_1, \frac{1}{y + h_1} Q_2), \ldots, (h_{\ell-1}, \frac{1}{y + h_{\ell-1}} Q_2)) \text{ if } i = 2$$

to get

$$T = \hat{e}(Q_1, Q_2)^{1/(y+h_0)} = \hat{e}(Q_1, Q_2)^{1/x} = \hat{e}(P_1, P_2)^{f^2(x)/x}.$$

7. Note that

$$\frac{1}{x}Q_2 = \frac{f(x)}{x}P_2 = \sum_{i=0}^{\ell-1} c_i x^{i-1} P_2 = c_0 \frac{1}{x} P_2 + \sum_{i=1}^{\ell-1} c_i x^{i-1} P_2.$$

Set

$$T' = \sum_{i=1}^{\ell-1} c_i x^{i-1} P_2 = \frac{f(x) - c_0}{x} P_2.$$

Then,

$$\hat{e}(\frac{1}{x}Q_1, Q_2) = \hat{e}(P_1, P_2)^{c_0^2/x} \cdot \hat{e}(\psi(T'), Q_2 + c_0 P_2).$$

Compute

$$\hat{e}(P_1, P_2)^{1/x} = (T/\hat{e}(\psi(T'), Q_2 + c_0 P_2))^{1/c_0^2}.$$

□

Sometimes, the decisional or the gap variants of above assumptions are used.

Assumption 2.6.8. ***(Decisional BDH (DBDH))*** *For $a, b, c, r \in_R \mathbb{Z}_p$, differentiating*

$$(aP_i, bP_j, cP_k, \hat{e}(P_1, P_2)^{abc}) \text{ and } (aP_i, bP_j, cP_k, \hat{e}(P_1, P_2)^r),$$

for some values of $i, j, k \in \{1, 2\}$, is hard.

Assumption 2.6.9. ***(Decisional BIDH (DBIDH))*** *For $a, b, r \in_R \mathbb{Z}_p$, differentiating*

$$(aP_i, bP_j, \hat{e}(P_1, P_2)^{b/a}) \text{ and } (aP_i, bP_j, \hat{e}(P_1, P_2)^r),$$

for some values of $i, j \in \{1, 2\}$, is hard.

Assumption 2.6.10. ***(Gap BDH (GBDH))*** *For $a, b, c, r \in_R \mathbb{Z}_p$, given (aP_i, bP_j, cP_k) for some values of $i, j, k \in \{1, 2\}$ and the algorithm of DBDH, computing $\hat{e}(P_1, P_2)^{abc}$ is hard.*

Assumption 2.6.11. ***(Gap BCAA1 (ℓ-GBCAA1))*** *For a positive integer ℓ, and $\alpha \in_R \mathbb{Z}_p^*$, given $(\alpha P_i, h_0, (h_1, \frac{1}{h_1+\alpha}P_j), \ldots, (h_\ell, \frac{1}{h_\ell+\alpha}P_j))$ for some values of $i, j \in \{1, 2\}$ where $h_i \in_R \mathbb{Z}_p^*$ and different from each other for $0 \leq i \leq \ell$, and the algorithm of DBIDH, computing $\hat{e}(P_1, P_2)^{1/(\alpha+h_0)}$ is hard.*

In the case where one has a computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, the existence of the pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$, implies that the DH assumption in $\mathbb{G}_2$ is in fact a gap assumption. We can use the pairing to construct an efficient decisional algorithm, which given (aP_2, bP_2, cP_2) returns 1 if $\hat{e}(\psi(aP_2), bP_2) = \hat{e}(P_1, cP_2)$, or 0 otherwise.

Whilst a particular scheme may not require the computable homomorphism to implement it, the computable homomorphism may be required in the security proof. If no such morphism exists, we are creating a relativised security proof, namely relative to an oracle which can compute ψ. We denote the corresponding relativised hard problem by a superscript-ψ, as in $\mathrm{DH}^{\psi}_{2,2,1}$, $\mathrm{BDH}^{\psi}_{2,1,2}$, ℓ-BDHI^{ψ}_{2}, ℓ-$\mathrm{BCAA1}^{\psi}_{1,2}$, etc.

To quantify the efficiency of an algorithm, we define (t, ϵ)-advantage of an algorithm for a computational problem as the probability ϵ of computing the correct value when the algorithm runs in time t, and define (t, ϵ)-advantage of an algorithm for a decisional problem with the value ϵ, which is the absolute value of the probability of correctly differentiating the input minus $1/2$, when the algorithm runs in time t.

2.7 Summary

In this chapter, we have introduced the pairing and the pairing-related computational complexity assumptions and the provable-security paradigm. Using pairing as the basic tool, in the following chapters we will construct a number of cryptographic schemes including efficient IBEs, CL-PKEs and identity-based two-party key agreement protocols. We will formally analyse the security of these schemes based on the pairing-related assumptions in the random oracle model.

Chapter 3

Efficient Identity-Based Encryption

Since Boneh-Franklin's pioneering work on identity-based encryption from pairings, there have been a number of IBEs proposed in the literature, which have proved security and have practical implementation. In this chapter, we present another secure IBE scheme: SK-IBE and then extend it to two identity-based key encapsulation mechanisms. These schemes make use of the Sakai-Kasahara's key construction and perform with the highest efficiency with regards to both computation and communication cost among the existing IBEs. We also investigate the impact of Cheon's algorithm of the strong Diffie-Hellman problem on the proposed schemes.

3.1 Introduction

To counter the impersonation attack, in the traditional public key cryptography, certificates are used to authenticate one's ownership of a claimed public key. And such a system has to include the infrastructure to issue certificates and one has to verify others' certificates to obtain authentic public keys. This type of system can be very complicated and difficult to manage and tedious to use.

In 1984 Shamir proposed a new public key cryptography paradigm: identity based cryptography (IBC) [138] to simplify the management of public keys. In IBC the public and private key pair is set up in a special way, i.e. the public key is the identifier (an

arbitrary string) of an entity, and the corresponding private key is created by using an identity-based key extraction algorithm, which binds the identifier with a master secret of a trusted authority, called the key generation center (KGC). As the identifier is used as the public key, *Bob* can immediately encrypt messages to *Alice* by using string "Alice" as the public key, hence a certificate is no longer necessary.

The IBC significantly simplifies the system construction and management, and authenticating the public key of an entity becomes trivial. However, it appears that the first public-key cryptography problem (see Section 2.1), i.e. designing the cryptographic primitive has become more difficult. Shamir presented an identity-based signature (IBS) scheme based on the RSA problem immediately after proposing the IBC notion [138]. But constructing a practical identity-based encryption (IBE) scheme had remained an open problem for many years.

An IBE scheme following Shamir's proposal consists of four algorithms:

1. **Setup** algorithm generates a master public/secret key pair.
2. **Extract** algorithm uses the master public/secret key pair to generate a user private key corresponding to an arbitrary identity string.
3. **Encrypt** algorithm encrypts a message using the master public key and the recipient identifier as the public key.
4. **Decrypt** algorithm decrypts the message using the user private key and the master public key.

The Setup and Extract algorithm together is called an ID key construction.

After nearly twenty years since Shamir proposed the notion, Cocks [42], Boneh and Franklin [19], and Sakai *et al.* [153] independently presented three IBE solutions in 2001. The Cocks solution is based on the quadratic residuosity problem. Both the Boneh-Franklin

solution and the Sakai *et al.* solution are based on the bilinear pairings on elliptic curves. Importantly Boneh and Franklin defined a well-formulated security model for IBE. Both pairing-based schemes are efficient in practice, and the Boneh-Franklin scheme (BF-IBE for short) received much attention owing to the fact that it was the first IBE scheme having a security proof in an appropriate model. After BF-IBE, there were several other IBE proposals constructed from pairings based on various complexity assumptions and with security proofs in Boneh-Franklin's model, e.g. [10, 12, 162, 82, 9].

In this chapter, we present three other IBE schemes, a full IBE in Section 3.4 and two ID-KEMs in Section 3.5. Using the Sakai-Kasahara ID key construction [151], these three schemes perform with the highest efficiency with regards to both computation and communication cost among the existing IBEs.

3.2 IBE Model

Following Shimir's proposal, we formally define the IBE. For an IBE scheme we define the message, ciphertext and randomness spaces by $\mathbb{M}_{\mathtt{ID}}(\cdot)$, $\mathbb{C}_{\mathtt{ID}}(\cdot)$ and $\mathbb{R}_{\mathtt{ID}}(\cdot)$. These spaces are parametrised by the master public key $M_{\mathfrak{pk}}$, and hence by the security parameter k. The scheme itself is specified by four polynomial time algorithms:

- **Setup** $\mathbb{G}_{\mathtt{ID}}(1^k)$: On input 1^k, the probabilistic algorithm outputs the master public key $M_{\mathfrak{pk}}$ and the master secret key $M_{\mathfrak{sk}}$.

$$(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID}}(1^k)$$

- **Extract** $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$: The probabilistic algorithm takes as the input $M_{\mathfrak{pk}}$, $M_{\mathfrak{sk}}$ and the identifier string $\mathtt{ID}_A \in \{0,1\}^*$ for entity A, and outputs the private key D_A associated with $\mathtt{ID}_A$.

$$D_A \leftarrow \mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$$

- **Encrypt** $\mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pt}}, \mathtt{ID}_A, m;\ r)$: The algorithm takes $M_{\mathfrak{pt}}, \mathtt{ID}_A$, the message $m \in \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pt}})$ and the randomness $r \in \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pt}})$ as the inputs, and outputs the ciphertext $C \in \mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pt}})$.

$$C \leftarrow \mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pt}}, \mathtt{ID}_A, m;\ r)$$

  We may also use the interface $\mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pt}}, \mathtt{ID}_A, m)$ by assuming that the random r is sampled in the algorithm.

- **Decrypt** $\mathbb{D}_{\mathtt{ID}}(M_{\mathfrak{pt}}, \mathtt{ID}_A, D_{\mathtt{ID}_A}, C)$: The deterministic algorithm takes $M_{\mathfrak{pt}}, \mathtt{ID}_A, D_A$ and C as input, and outputs the plaintext m or a failure symbol $\perp$ if C is invalid.

$$(m \text{ or } \perp) \leftarrow \mathbb{D}_{\mathtt{ID}}(M_{\mathfrak{pt}}, \mathtt{ID}_A, D_A, C)$$

To cope with probabilistic ciphers, we will require that not too many choices for r encrypt a given message to a given ciphertext. To formalize this concept we let $\gamma(M_{\mathfrak{pt}})$ be the least upper bound such that

$$\mid \{r \in \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pt}}) : \mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pt}}, \mathtt{ID}, m;\ r) = C\} \mid \leq \gamma(M_{\mathfrak{pt}})$$

for every $\mathtt{ID}$, $m \in \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pt}})$ and $C \in \mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pt}})$. Our requirement is that the quantity $\gamma = \frac{\gamma(M_{\mathfrak{pt}})}{|\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pt}})|}$ is a negligible function of the security parameter.

Following the Boneh and Franklin IBE security formulation [19] we can define various security notions for an IBE scheme. All are based on one of the two-stage games in Table 3.1 between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of the encryption algorithm and a challenger.

In the games, s is some state information and $\mathcal{O}_{\mathtt{ID}}$ are oracles to which the adversary has access. There are various possibilities for these oracles depending on the attack model:

- CPA Model: In this model the adversary only has access to a private key extraction oracle, which on input of $\mathtt{ID} \neq \mathtt{ID}^*$ will output the corresponding value of $D_{\mathtt{ID}}$.

Table 3.1: IBE Security Formulation

ID-OW Adversarial Game	ID-IND Adversarial Game
1. $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID}}(1^k)$.	1. $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID}}(1^k)$.
2. $(s, \mathtt{ID}^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathtt{pk}})$.	2. $(s, \mathtt{ID}^*, m_0, m_1) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathtt{pk}})$.
3. $m \leftarrow \mathbb{M}_{\mathtt{ID}}(M_{\mathtt{pk}}), r \leftarrow \mathbb{R}_{\mathtt{ID}}(M_{\mathtt{pk}})$.	3. $b \leftarrow \{0,1\}, r \leftarrow \mathbb{R}_{\mathtt{ID}}(M_{\mathtt{pk}})$.
4. $C^* \leftarrow \mathbb{E}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}^*, m;\ r)$.	4. $C^* \leftarrow \mathbb{E}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}^*, m_b;\ r)$.
5. $m' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathtt{pk}}, C^*, s, \mathtt{ID}^*)$.	5. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathtt{pk}}, C^*, s, \mathtt{ID}^*, m_0, m_1)$.

- CCA2 Model: In this model the adversary has access to the private key extraction oracle as above and it also has access to a decryption oracle with respect to any identity ID and ciphertext C of its choice. The decryption oracle outputs the decryption of C using the private key associated with ID. There is one restriction on how the adversary uses this oracle: In the second phase $\mathcal{A}_2$ is not allowed to query the decryption oracle with the pair $(\mathtt{ID}^*, C^*)$.

Let MOD denote the mode of attack, either CPA or CCA2. The adversary's advantage in the first game is defined to be

$$\mathtt{Adv}_{\mathtt{ID}}^{\mathtt{ID\text{-}OW\text{-}MOD}}(\mathcal{A}) = \Pr[m' = m],$$

while, the advantage in the second game is given by

$$\mathtt{Adv}_{\mathtt{ID}}^{\mathtt{ID\text{-}IND\text{-}MOD}}(\mathcal{A}) = \mid 2\Pr[b' = b] - 1 \mid .$$

An IBE algorithm is considered to be secure, in the sense of a given goal and attack model (ID-IND-CCA2 for example) if, for all PPT adversaries, the advantage in the relevant game is a negligible function of the security parameter k.

In [56], Canetti *et al.* proposed a weaker security model, called selective identity IBE. In this model, the adversary has to commit to the identity $\mathtt{ID}^*$ that it intends to attack at the

beginning of the game, even before it is given the system parameters. Similar to the above ID-IND game, we can define the security models of sID-IND-CPA and sID-IND-CCA2.

3.3 Several Influential Key Constructions in IBE

After Cocks and Boneh-Franklin's pioneering works, based on various complexity assumptions there are a number of IBE schemes constructed with or without random oracles. In [9], the exiting IBEs are categorised into four classes. Following this categorisation, we recall an influential ID key construction in each class. Some of these key constructions have been used in other identity-based cryptographic mechanisms.

Quadratic residuosity IBE. In 2001, Cocks published an IBE [42], which does not use pairings. The scheme uses the following key construction.

- **Setup**. Randomly sample two primes p_1, p_2 with $p_1, p_2 = 3 \mod 4$ as the master secret key and publish $N = p_1 \cdot p_2$ as the master public key.
- **Extract**. Given an identifier $\mathtt{ID} \in \{0,1\}^*$, compute the associated private key as

$$D_{\mathtt{ID}} = \begin{cases} H(\mathtt{ID})^{\frac{N-(p_1+p_2)+5}{8}} \mod N, & \text{if } H(ID) \text{ is a square mod } N \\ (-H(\mathtt{ID}))^{\frac{N-(p_1+p_2)+5}{8}} \mod N, & \text{if } -H(ID) \text{ is a square mod } N \end{cases}$$

  where H is universal hash function with the codomain J_N defined by

$$J_N = \{a \mid a \in \mathbb{Z}_N \text{ and Jacobi symbol } (\frac{a}{N}) = 1\}.$$

The key construction relies on the hardness of the square root problem, i.e. given a composite modulus $N = p_1 \cdot p_2$ (a Blum number) and a quadratic residue $x \in \mathbb{Z}_N$, computing y with $y^2 = x \mod N$. We note that the security of the full Cocks' IBE relies on the quadratic residuosity assumption, i.e. given a composite modulus $N = p_1 \cdot p_2$ and a modular residue $x \in \mathbb{Z}_N$, determining if x is a square is hard. Cocks' IBE is not very efficient with communication because it has big ciphertext expansion: For an ℓ-bit message, the ciphertext is

$\ell \log_2 N$-bit long if $H(\mathtt{ID})$ is known to be a square modulo N, otherwise the ciphertext size has to be doubled.

Full domain hash IBE with pairings. In 2000, Sakai *et al.* published an identity-based non-interactive key establishment with pairings [152]. In this work, the authors proposed an ID key construction, which we call the *SOK ID key construction*, working as follows.

- **Setup**. Randomly sample $s \in \mathbb{Z}_p$ as the master secret key and publish (P, sP) with other parameters as the master public key, where P is a generator of a pairing group $\mathbb{G}$ of order p on an elliptic curve.

- **Extract**. Given an identifier $\mathtt{ID} \in \{0,1\}^*$, compute the associated private key $D_{\mathtt{ID}} = sH(\mathtt{ID})$, where H is a uniformly distributed hash function with codomain $\mathbb{G}$, i.e. $H : \{0,1\}^* \rightarrow \mathbb{G}$.

The construction relies on the DH assumption. Later the same approach was adopted independently by Boneh-Franklin and Sakai *et al.* to construct IBEs [19, 153]. This ID key construction has been widely used in other identity-based cryptographic mechanisms such as the IBS [93, 45] and the IB-KAPs [143, 63, 161]. The full domain hash method has a shortcoming: The operation to embed an arbitrary string on the chosen pairing group could be expensive; for some pairing groups, the operation could be too costly to be applied in a practical implementation.

Exponent inversion IBE with pairings. In 2003, Sakai and Kasahara published another category of identity-based cryptographic schemes [151]. These schemes make use of a different key construction, which we call the *SK ID key construction*, and the idea can be tracked back to the traitor tracing work in 2002 [128]. Instead of using the expensive full domain hash method, the SK key construction works as follows:

- **Setup**. Randomly sample an integer $s \in \mathbb{Z}_p^*$ as the master secret key and publish (P, sP) with other parameters as the master public key, where P again is a generator of a pairing group $\mathbb{G}$ of order p.

- **Extract**. Given an identifier string $\mathtt{ID} \in \{0,1\}^*$, generate the associated private key as $D_{\mathtt{ID}} = \frac{1}{(s+H(\mathtt{ID}))}P$, where H is a universal hash function with codomain $\mathbb{Z}_p^*$.

The key construction relies on the hardness assumption that given (P, xP), computing the inversion $1/xP$ is hard, or more accurately the ℓ-SDH assumption. Later some variants of the construction were used in other IBEs [10, 82]. The SK key construction has been used in other identity-based primitives including IBS [25] and ID key agreements [125]. The security of these schemes rely on a stronger assumption ℓ-BDHI or its variant.

Commutative blinding IBE with pairings. To demonstrate the constructibility of IBE in the standard model (without random oracles), in [12] Boneh and Boyen presented a selective identity IBE [56] using a new approach to constructing the Setup and Extract algorithm, which Boyen referred to as "commutative blinding" [9]. Later, Waters used a variant of the key construction to fully solve the problem of constructing ID-IND-CCA2 secure IBE without random oracles. The Water's ID key construction works as follows:

- **Setup**. First randomly sample $\alpha \in Z_p, P \in \mathbb{G}, P_2 \in \mathbb{G}$ and set αP_2 as the master secret key. Next, pick n random elements $U_i \in \mathbb{G}$ to form an n-length vector $\vec{U} =< U_i >$ and another random $U' \in \mathbb{G}$ and compute $P_1 = \alpha P$. Publish $(P, P_1, P_2, U', \vec{U})$ as the master public key.

- **Extract**. For an identifier $\mathtt{ID} \in \{0,1\}^n = (\delta_1 \cdots \delta_n)$ with $\delta_i \in \{0,1\}$, randomly choose $r \in \mathbb{Z}_p$ and output the private key

$$D_{\mathtt{ID}} = (\alpha P_2 + r(U' + \sum_{i=1}^{n} \delta_i U_i), rP).$$

This key construction has been used in other schemes such as IBEs [66, 131, 108] and IBS [136]. Recently, Boyen proposed another variant of the key construction but using a hash function H modeled as a random oracle on an arbitrary identifier string [9] (see Section 4.6.2 for details).

3.4 SK-IBE

In this section, we present an IBE using Sakai-Kasahara ID key construction and analyse its security. We choose the simplest variant of the Sakai-Kasahara IBE scheme [151] as the basic version of SK-IBE. To achieve security against adaptive chosen ciphertext attacks, we make use of the Fujisaki-Okamoto transformation [73] in the same way that it was used in BF-IBE [19].

3.4.1 The Scheme

Following the formulation defined in Section 3.2, SK-IBE is specified by four polynomial time algorithms:

Setup $\mathbb{G}_{\mathrm{ID}}(1^k)$. On input 1^k, the algorithm works as follows:

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ of prime order p, an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$. Pick a random generator $P_2 \in \mathbb{G}_2^*$ and set $P_1 = \psi(P_2)$.

2. Pick a random $s \in \mathbb{Z}_p^*$ and compute $P_{pub} = sP_1$.

3. Pick four cryptographic hash functions

$$
\begin{aligned}
H_1 &: \{0,1\}^* \rightarrow \mathbb{Z}_p^*, \\
H_2 &: \mathbb{G}_t \rightarrow \{0,1\}^n, \\
H_3 &: \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_p^*, \\
H_4 &: \{0,1\}^n \rightarrow \{0,1\}^n,
\end{aligned}
$$

   for some integer $n > 0$.

4. Output the master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, H_1, H_2, H_3, H_4)$ and the master secret key $M_{\mathfrak{sk}} = s$.

The message space is $\mathbb{M} = \{0,1\}^n$, the ciphertext space is $\mathbb{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$, and the randomness space is $\mathbb{R} = \{0,1\}^n$.

Extract $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$**.** Given an identifer string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathfrak{pk}}$ and $M_{\mathfrak{sk}}$, the algorithm returns $D_A = \frac{1}{s+H_1(\mathtt{ID}_A)} P_2$.

Encrypt $\mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, m)$**.** Given a message $m \in \mathbb{M}$, $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, m)$.
2. Compute $Q_A = H_1(\mathtt{ID}_A)P_1 + P_{pub}$ and $g^r = \hat{e}(P_1, P_2)^r$.
3. Set the ciphertext to $C = \langle rQ_A, \sigma \oplus H_2(g^r), m \oplus H_4(\sigma) \rangle$.

Decrypt $\mathbb{D}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, D_{\mathtt{ID}_A}, C)$**.** Given a ciphertext $C = \langle U, V, W \rangle \in \mathbb{C}$, $\mathtt{ID}_A$, D_A and $M_{\mathfrak{pk}}$, follow the steps:

1. Compute $g^{r'} = \hat{e}(U, D_A)$ and $\sigma' = V \oplus H_2(g^{r'})$.
2. Compute $m' = W \oplus H_4(\sigma')$ and $r' = H_3(\sigma', m')$.
3. Compute $Q_A = H_1(\mathtt{ID}_A)P_1 + P_{pub}$. If $U \neq r'Q_A$, output $\perp$, else return m' as the plaintext.

A few optimisations can be applied in SK-IBE. (1) In the Encrypt algorithm, the pairing $g = \hat{e}(P_1, P_2)$ is fixed and can be pre-computed. It can further be treated as a system public parameter. Therefore, no pairing computation is required in Encrypt. If the optimization of fixed-based multiplication in $\mathbb{G}_1$ is available, then rQ_A can be computed as $rH_1(\mathtt{ID}_A)P_1 + rP_{pub}$. (2) In the Decrypt algorithm, the last step involves the operation to compute the real public key of the decrypting party $Q_A = H(\mathtt{ID}_A)P_1 + P_{pub}$, which can be pre-computed to

save a multiplication in $\mathbb{G}_1$. (3) The isomorphism ψ is not required to implement the scheme. Hence P_1 can be randomly sampled from $\mathbb{G}_1^*$ instead and we know that an isomorphism which may not be efficiently computable always exists. As the schemes does not require a hash operation to neither $\mathbb{G}_1$ nor $\mathbb{G}_2$, the scheme is very flexible and can be implemented with all four types of pairing defined in Section 2.5.3.

3.4.2 Security Analysis of SK-IBE

Now we evaluate the security of SK-IBE. We prove that the security of SK-IBE can reduce to the hardness of the ℓ-BCAA1 problem (so the ℓ-BDHI problem by Theorem 2.6.3).

Theorem 3.4.1. *SK-IBE is secure against ID-IND-CCA2 adversaries provided that $H_i (1 \leq i \leq 4)$ are random oracles and the ℓ-BCAA1 assumption is sound. Specifically, suppose there exists an ID-IND-CCA2 adversary $\mathcal{A}$ against SK-IBE that has advantage $\epsilon(k)$ and running time $t(k)$. Suppose also that during the attack, $\mathcal{A}$ makes at most q_D decryption queries and at most q_i queries on H_i for $1 \leq i \leq 4$ respectively. Then there exists an algorithm $\mathcal{B}$ to solve the $(q_1 - 1)$-$BCAA1_{1,2}$ problem with advantage $Adv_{\mathcal{B}}(k)$ and running time $t_{\mathcal{B}}(k)$ where*

$$\begin{aligned} \boldsymbol{Adv}_{\mathcal{B}}^{(q_1-1)-\boldsymbol{BCAA1}_{1,2}}(k) &\geq \tfrac{1-q_2/2^n}{2(q_3+q_4)}[(\tfrac{\epsilon(k)}{q_1}+1)(1-\tfrac{2}{p})^{q_D}-1], \\ t_{\mathcal{B}}(k) &\leq t(k) + O((q_3+q_4)\cdot(n+\log_2 p)), \end{aligned}$$

where p is the order of $\mathbb{G}_1$ and n is the length of σ. Note that H_1 can be queried directly by $\mathcal{A}$ or indirectly by an extraction query, a decryption query or the challenge operation.

Proof: The theorem follows immediately by combining Lemma 3.4.2, 3.4.3 and 3.4.4. The reduction with three steps can be sketched as follows. First, we prove that if there exists an ID-IND-CCA2 adversary, who is able to break SK-IBE by launching the adaptive chosen ciphertext attacks as defined in the security model of Section 3.2, then there exists an IND-CCA2 adversary to break the $\mathbf{BasicPub}^{\mathbf{hy}}$ scheme defined in Lemma 3.4.2. Second, if such an IND-CCA2 adversary exists, then we show (in Lemma 3.4.3) that there must be an OW-CPA adversary that breaks the corresponding **BasicPub** scheme. Finally, in Lemma 3.4.4 we prove that if the **BasicPub** scheme is not secure against an OW-CPA adversary, then the corresponding ℓ-BCAA1 assumption is flawed. □

Lemma 3.4.2. *Suppose that H_1 is a random oracle and that there exists an ID-IND-CCA2 adversary $\mathcal{A}$ against SK-IBE with advantage $\epsilon(k)$ which makes at most q_1 distinct queries to H_1. Then there exists an IND-CCA2 adversary $\mathcal{B}$ which runs in time $O(time(\mathcal{A}))$ against the following $\boldsymbol{BasicPub}^{\boldsymbol{hy}}$ scheme with advantage at least $\epsilon(k)/q_1$. Note that H_1 can be queried directly by $\mathcal{A}$ or indirectly by an extraction query, a decryption query or the challenge operation.*

*$\boldsymbol{BasicPub}^{\boldsymbol{hy}}$ is specified by three algorithms: **keygen**, **encrypt** and **decrypt**.*

***keygen:** On input 1^k, the parameter generator takes the following steps.*

1. *Identical with Step 1 in Setup algorithm of SK-IBE.*

2. *Pick a random* $s \in \mathbb{Z}_p^*$ *and compute* $P_{pub} = sP_1$. *Randomly choose different elements* $h_i \in \mathbb{Z}_p^*$ *and compute* $\frac{1}{h_i+s}P_2$ *for* $0 \leq i < q_1$.

3. *Pick three cryptographic hash functions:* $H_2 : \mathbb{G}_t \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_p^*$ *and* $H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$ *for some integer* $n > 0$.

The message space is $\mathbb{M} = \{0,1\}^n$, *the ciphertext space is* $\mathbb{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$ *and the randomness space is* $\mathbb{R} = \{0,1\}^n$. *The public key is* $K_{pub} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, h_0, (h_1, \frac{1}{h_1+s}P_2), \ldots, (h_i, \frac{1}{h_i+s}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2), H_2, H_3, H_4)$ *and the private key is* $K_{priv} = \frac{1}{h_0+s}P_2$. *Note that* $\hat{e}(h_0P_1 + P_{pub}, K_{priv}) = \hat{e}(P_1, P_2)$.
encrypt: *Given a plaintext* $m \in \mathbb{M}$ *and the public key* K_{pub}, *the algorithm proceeds as follows:*

1. *Pick a random* $\sigma \in \{0,1\}^n$ *and compute* $r = H_3(\sigma, m)$, *and* $g^r = \hat{e}(P_1, P_2)^r$.

2. *Set the ciphertext to* $C = \langle r(h_0P_1 + P_{pub}), \sigma \oplus H_2(g^r), m \oplus H_4(\sigma)\rangle$.

decrypt: *Given a ciphertext* $C = \langle U, V, W\rangle$, K_{pub}, *and the private key* K_{priv}, *the algorithm takes the following steps.*

1. *Compute* $g^{r'} = \hat{e}(U, K_{priv})$ *and* $\sigma' = V \oplus H_2(g^{r'})$.

2. *Compute* $m' = W \oplus H_4(\sigma')$ *and* $r' = H_3(\sigma', m')$.

3. *If* $U \neq r'(h_0P_1 + P_{pub})$, *reject the ciphertext, else return* m' *as the plaintext.*

Proof: We construct an IND-CCA2 adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage against $\mathbf{BasicPub^{hy}}$. The game between a challenger $\mathcal{C}$ and the adversary $\mathcal{B}$ starts with the challenger first generating a random public key K_{pub} by running algorithm keygen of $\mathbf{BasicPub^{hy}}$ ($\log_2 q_1$ is part of the security parameter of $\mathbf{BasicPub^{hy}}$). The result is $K_{pub} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, h_0, (h_1, \frac{1}{h_1+s}P_2), \ldots, (h_i, \frac{1}{h_i+s}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2), H_2, H_3, H_4)$, where $P_{pub} = sP_1$ with $s \in \mathbb{Z}_p^*$, and the private key $K_{priv} = \frac{1}{h_0+s}P_2$. The challenger passes K_{pub} to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ mounts an IND-CCA2 attack on the $\mathbf{BasicPub^{hy}}$ scheme with the public key K_{pub} with the help of $\mathcal{A}$ as follows.

$\mathcal{B}$ randomly chooses an index I with $1 \leq I \leq q_1$ and simulates the algorithm Setup of SK-IBE for $\mathcal{A}$ by supplying $\mathcal{A}$ with the SK-IBE master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, H_1, H_2, H_3, H_4)$ where H_1 is a random oracle controlled by $\mathcal{B}$. The master secret key $M_{\mathfrak{sk}}$ for this cryptosystem is s, although $\mathcal{B}$ does not know this value. Adversary $\mathcal{A}$ can make queries on H_1 at any time.

- $H_1(\mathtt{ID}_i)$: $\mathcal{B}$ maintains a list of tuples $(\mathtt{ID}_i, h_i, d_i)$ indexed by $\mathtt{ID}_i$ as explained below. We refer to this list as H_1^{list} which is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point $\mathtt{ID}_i$, $\mathcal{B}$ responds as follows:

- If $\mathtt{ID}_i$ already appears on the H_1^{list} in a tuple $(\mathtt{ID}_i, h_i, d_i)$, then $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = h_i$.
- Otherwise, if the query is on the I-th distinct ID and $\perp$ is not used as d_i (this could be inserted by the challenge operation specified later) by any existing tuple, then $\mathcal{B}$ stores $(\mathtt{ID}_I, h_0, \perp)$ into the tuple list and responds with $H_1(\mathtt{ID}_I) = h_0$.
- Otherwise, $\mathcal{B}$ selects a random integer $h_i (i > 0)$ from K_{pub} which has not been chosen by $\mathcal{B}$ and stores $(\mathtt{ID}_i, h_i, \frac{1}{h_i+s}P_2)$ into the tuple list. $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = h_i$.

- **Phase 1:** $\mathcal{A}$ launches Phase 1 of its attack, by making a series of requests, each of which is either an extraction or a decryption query. $\mathcal{B}$ replies to these requests as follows.
  - **Extraction**$(\mathtt{ID}_i)$**:** $\mathcal{B}$ first looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, then $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. $\mathcal{B}$ then checks the value d_i: if $d_i \neq \perp$, $\mathcal{B}$ responds with d_i; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).
  - **Decryption**$(\mathtt{ID}_i, C_i)$**:** $\mathcal{B}$ first looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, then $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. If $d_i = \perp$, then $\mathcal{B}$ sends the decryption query $C_i = \langle U, V, W \rangle$ to $\mathcal{C}$ and simply relays the plaintext got from $\mathcal{C}$ to $\mathcal{A}$ directly. Otherwise, $\mathcal{B}$ decrypts the ciphertext by first computing $g^{r'} = \hat{e}(U, d_i)$, then querying $\zeta = H_2(g^{r'})$ (H_2 is controlled by $\mathcal{C}$), and computing $\sigma' = V \oplus \zeta, m' = W \oplus H_4(\sigma')$ and $r' = H_3(\sigma', m')$. Finally $\mathcal{B}$ checks the validity of C_i as Step 3 of algorithm decrypt and returns m', if C_i is valid, otherwise the failure symbol $\perp$.

- **Challenge:** At some point, $\mathcal{A}$ decides to end Phase 1 and picks $\mathtt{ID}^*$ and two messages (m_0, m_1) on which it wants to be challenged. Based on the queries on H_1 so far, $\mathcal{B}$ responds differently.
  - If the I-th query on H_1 has been issued,
    - if $\mathtt{ID}_I = \mathtt{ID}^*$ (and so $D_{\mathtt{ID}^*} = \perp$), $\mathcal{B}$ continues;
    - otherwise, $\mathcal{B}$ aborts the game (**Event 2**).
  - Otherwise,
    - if the tuple corresponding to $\mathtt{ID}^*$ is on list H_1^{list} (and so $D_{\mathtt{ID}^*} \neq \perp$), then $\mathcal{B}$ aborts the game (**Event 3**);
    - otherwise, $\mathcal{B}$ inserts the tuple $(\mathtt{ID}^*, h_0, \perp)$ into the list and continues (this operation is treated as an H_1 query in the simulation).

  Note that after this point, it must have $H_1(\mathtt{ID}^*) = h_0$ and $D_{\mathtt{ID}^*} = \perp$. $\mathcal{B}$ passes $\mathcal{C}$ the pair (m_0, m_1) as the messages on which it wishes to be challenged. $\mathcal{C}$ randomly chooses $b \in \{0, 1\}$, encrypts m_b and responds with the ciphertext $C^* = \langle U', V', W' \rangle$. Then $\mathcal{B}$ forwards C^* to $\mathcal{A}$.

- **Phase 2:** $\mathcal{B}$ continues to respond to requests in the same way as it did in Phase 1. Note that following the rules, the adversary will not issue the extraction query on $\mathtt{ID}^*$ (for which $D_{\mathtt{ID}^*} = \perp$) and the decryption query on $(\mathtt{ID}^*, C^*)$. And so, $\mathcal{B}$ always can answer other queries without aborting the game.

- **Guess:** $\mathcal{A}$ makes a guess b' for b. $\mathcal{B}$ outputs b' as its own guess.

Claim: If the algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}$'s view is identical to its view in the real attack.

Proof: $\mathcal{B}$'s responses to H_1 queries are uniformly and independently distributed in $\mathbb{Z}_p^*$ as in the real attack because of the behavior of algorithm keygen in the $\mathbf{BasicPub}^{\mathbf{hy}}$ scheme. All responses to $\mathcal{A}$'s requests are valid, if $\mathcal{B}$ does not abort. Furthermore, the challenge ciphertext $C^* = \langle U', V', W' \rangle$ is a valid encryption in SK-IBE for m_b where $b \in \{0, 1\}$ is random.

The remaining problem is to estimate the probability that $\mathcal{B}$ does not abort during simulation. Algorithm $\mathcal{B}$ could abort when one of the following events happens: (1) **Event 1**: $\mathcal{A}$ queried a private key which is represented by $\perp$ at some point. Recall that only one private key is represented by $\perp$ in the whole simulation which could be inserted in an H_1 query (as the private key of $\mathtt{ID}_I$) in Phase 1 or in the challenge phase (as the private key of $\mathtt{ID}^*$). Because of the rules of the game, the adversary will not query the private key of $\mathtt{ID}^*$. Hence, this event happens only when the adversary extracted the private key of $\mathtt{ID}_I \neq \mathtt{ID}^*$, meanwhile $d_I = \perp$, i.e. $\mathtt{ID}_I \neq \mathtt{ID}^*$ and $H_1(\mathtt{ID}_I)$ was queried in Phase 1; (2) **Event 2**: the adversary wants to be challenged on an identity $\mathtt{ID}^* \neq \mathtt{ID}_I$ and $H_1(\mathtt{ID}_I)$ was queried in Phase 1; (3) **Event 3**: the adversary wants to be challenged on an identity $\mathtt{ID}^* \neq \mathtt{ID}_I$ and $H_1(\mathtt{ID}_I)$ was queried in Phase 2.

Notice that all the three events imply **Event 4** that the adversary did not choose $\mathtt{ID}_I$ as the challenge identity. Hence we have

$$\Pr[\mathcal{B} \text{ does not abort}] = \Pr[\overline{\texttt{Event 1}} \wedge \overline{\texttt{Event 2}} \wedge \overline{\texttt{Event 3}}] \geq \Pr[\overline{\texttt{Event 4}}] \geq 1/q_1.$$

So, the lemma follows. □

Lemma 3.4.3. *Let H_3, H_4 be random oracles. Let $\mathcal{A}$ be an IND-CCA2 adversary against* $\mathbf{BasicPub}^{\mathbf{hy}}$ *defined in Lemma 3.4.2 with advantage $\epsilon(k)$. Suppose $\mathcal{A}$ has running time $t(k)$, makes at most q_D decryption queries, and makes q_3 and q_4 queries to H_3 and H_4 respectively. Then there exists an OW-CPA adversary $\mathcal{B}$ against the following* **BasicPub** *scheme, which is specified by three algorithms:***keygen**, **encrypt** *and* **decrypt**.
keygen: *On input 1^k, the parameter generator follows the steps.*

1. *Identical with step 1 in algorithm keygen of* $\mathbf{BasicPub}^{\mathbf{hy}}$.
2. *Identical with step 2 in algorithm keygen of* $\mathbf{BasicPub}^{\mathbf{hy}}$.
3. *Pick a cryptographic hash function $H_2 : \mathbb{G}_t \to \{0,1\}^n$ for some integer $n > 0$.*

The message space is $\mathbb{M} = \{0,1\}^n$, *the ciphertext space is* $\mathbb{C} = \mathbb{G}_1^* \times \{0,1\}^n$ *and the randomness space is* $\mathbb{R} = \mathbb{Z}_p^*$. *The public key is* $K_{pub} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, h_0, (h_1, \frac{1}{h_1+s}P_2), \ldots, (h_i, \frac{1}{h_i+s}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2), H_2)$ *and the private key is* $K_{priv} = \frac{1}{h_0+s}P_2$. *Again it has* $\hat{e}(h_0P_1 + P_{pub}, K_{priv}) = \hat{e}(P_1, P_2)$.

encrypt: *Given a plaintext* $m \in \mathbb{M}$ *and the public key* K_{pub}, *choose a random* $r \in \mathbb{Z}_p^*$ *and compute ciphertext* $C = \langle r(h_0P_1 + P_{pub}), m \oplus H_2(g^r)\rangle$ *where* $g^r = \hat{e}(P_1, P_2)^r$.

decrypt: *Given a ciphertext* $C = \langle U, V\rangle$, K_{pub}, *and the private key* K_{priv}, *compute* $g^{r'} = \hat{e}(U, K_{priv})$ *and plaintext* $m = V \oplus H_2(g^{r'})$.

with advantage $\epsilon_1(k)$ and running time $t_1(k)$ where

$$\epsilon_1(k) \geq \frac{1}{2(q_3+q_4)}[(\epsilon(k)+1)(1-\frac{2}{p})^{q_D} - 1],$$
$$t_1(k) \leq t(k) + O((q_3+q_4)\cdot(n+\log_2 p)),$$

where p is the order of $\mathbb{G}_1$ and n is the length of σ.

Proof: This lemma follows from the result of the Fujisaki-Okamoto transformation [73]. □

Lemma 3.4.4. *Let* H_2 *be a random oracle. Suppose there exists an OW-CPA adversary* $\mathcal{A}$ *against the* ***BasicPub*** *defined in Lemma 3.4.3 which has advantage* $\epsilon(k)$ *and queries* H_2 *at most* q_2 *times. Then there exists an algorithm* $\mathcal{B}$ *to solve the* (q_1-1)*-BCAA1*$_{1,2}$ *problem with advantage* $(1 - q_2/2^n)\epsilon(k)$ *and running time* $O(time(\mathcal{A}))$.

Proof: Algorithm $\mathcal{B}$ is given as input a random (q_1-1)-BCAA1$_{1,2}$ instance $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, P_1, P_2, xP_1, h_0, (h_1, \frac{1}{h_1+x}P_2), \ldots, (h_i, \frac{1}{h_i+x}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+x}P_2))$ where x is a random element from $\mathbb{Z}_p^*$. Algorithm $\mathcal{B}$ finds $\hat{e}(P_1, P_2)^{1/x}$ by interacting with $\mathcal{A}$ as follows:

Algorithm $\mathcal{B}$ first simulates algorithm keygen of **BasicPub** by setting $K_{pub} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, P_1, P_2, xP_1, h_0, (h_1, \frac{1}{h_1+x}P_2), \ldots, (h_i, \frac{1}{h_i+x}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+x}P_2), H_2)$, i.e. setting $P_{pub} = xP_1$ and H_2 is the random oracle controlled by $\mathcal{B}$. Hence $K_{priv} = \frac{1}{h_0+x}P_2$. Now $\mathcal{B}$ starts to respond to queries as follows.

- $H_2(X_i)$: At any time algorithm $\mathcal{A}$ can issue queries to the random oracle H_2. To respond to these queries $\mathcal{B}$ maintains a list H_2^{list}. Each entry in the list is a tuple of the form (X_i, ζ_i) indexed by X_i. To respond to a query on X_i, $\mathcal{B}$ does the following operations:
  - If on the list there is a tuple indexed by X_i, then $\mathcal{B}$ responds with ζ_i.
  - Otherwise, $\mathcal{B}$ randomly chooses a string $\zeta_i \in \{0,1\}^n$ and inserts a new tuple (X_i, ζ_i) into the list. It responds to $\mathcal{A}$ with ζ_i.

- **Challenge:** $\mathcal{B}$ chooses a random string $V^* \in \{0,1\}^n$ and a random element $r \in \mathbb{Z}_p^*$ and computes $U^* = rP_1$. $\mathcal{B}$ sets $C^* = \langle U^*, V^* \rangle$ and gives C^* as the challenge to $\mathcal{A}$. Observe that the decryption of C^* is

$$m^* = V^* \oplus H_2(\hat{e}(U^*, K_{priv})) = V^* \oplus H_2(\hat{e}(rP_1, \frac{1}{x+h_0}P_2)).$$

- **Guess:** After algorithm $\mathcal{A}$ outputs its guess m', $\mathcal{B}$ responds to the q_1-BCAA1 challenge as follows:
  - $\mathcal{B}$ computes $\zeta^* = m' \oplus V^*$.
  - $\mathcal{B}$ goes through H_2^{list} to find tuples (X_i, ζ^*).
  - If no tuples or more than one tuple is found, $\mathcal{B}$ fails the game (**Event 1**).
  - $\mathcal{B}$ returns $T = X_i^{1/r}$ as the response to the q_1-BCAA1 challenge.

`Event 1` implies that $X^* = \hat{e}(rP_1, \frac{1}{x+h_0}P_2)$ has not been queried on H_2 at some point during the simulation above. As H_2 is a random oracle, if `Event 1` happens, $m^* = m'$ can only happen with negligible probability $1/2^n$. Again as H_2 is a random oracle, $H_2(X') = H_2(X^*) = \zeta^*$ for $X' \neq X^*$ in the simulation happens with probability at most $q_2/2^n$.

Overall, we have

$$\Pr[\mathcal{B} \text{ wins}] \approx (1 - q_2/2^n)\epsilon(k).$$

This completes the proof of Theorem 3.4.1. □

3.5 SK-KEM

SK-IBE like many other schemes such as BF-IBE is suitable for encrypting short messages. A natural way to process arbitrarily long messages is to use *hybrid encryption.* A hybrid encryption scheme consists of two basic operations. One operation uses a public-key encryption technique (a so-called *key encapsulation mechanism* or KEM) to derive and encrypt a shared key; the other operation uses the shared key in a symmetric-key algorithm (a so-called *data encapsulation mechanism* or DEM) to encrypt the actual message. Cramer and Shoup [65] formalised the notion of hybrid encryption and presented sufficient conditions for a KEM and a DEM to construct IND-CCA2 secure public key encryption. Recently, Bentahar *et al.* [20] extended the KEM concept to the identity-based setting. The idea is to construct an ID-IND-CCA2 secure IBE scheme from an ID-IND-CCA2 secure ID-KEM and a secure DEM.

In this section we present two efficient ID-KEMs using the SK ID key construction. We first present a simple ID-KEM: SK-KEM1, with a succinct construction as ECIES-KEM [101]. However, like ECIES-KEM the scheme has to base its security on a gap assumption: ℓ-GBCAA1. To recover from this weakness, we further construct another ID-KEM: SK-KEM2. SK-KEM2 is slightly slower than SK-KEM1 and has a larger ciphertext, but bases its security on the ℓ-BCAA1 assumption as SK-IBE.

3.5.1 Identity-Based Key Encapsulation Mechanism

Before presenting concrete schemes we first formally describe ID-KEM and DEM, and then introduce the combination theory that a full IBE scheme can be constructed by combining an ID-KEM with a DEM.

For an ID-KEM scheme we define the key, encapsulation and randomness spaces by $\mathbb{K}_{\mathtt{ID\text{-}KEM}}(\cdot)$, $\mathbb{C}_{\mathtt{ID\text{-}KEM}}(\cdot)$ and $\mathbb{R}_{\mathtt{ID\text{-}KEM}}(\cdot)$. These spaces are parametrised by the master public key $M_{\mathtt{pk}}$, and hence by the security parameter k. The ID-KEM scheme is specified by four polynomial time algorithms:

- **Setup** $\mathbb{G}_{\mathtt{ID\text{-}KEM}}(1^k)$: The algorithm is the same as $\mathbb{G}_{\mathtt{ID}}(1^k)$.
- **Extract** $\mathbb{X}_{\mathtt{ID\text{-}KEM}}(M_{\mathtt{pk}}, M_{\mathtt{sk}}, \mathtt{ID}_A)$: The algorithm is the same as $\mathbb{X}_{\mathtt{ID}}(M_{\mathtt{pk}}, M_{\mathtt{sk}}, \mathtt{ID}_A)$.
- **Encapsulate** $\mathbb{E}_{\mathtt{ID\text{-}KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A;\ r)$: This algorithm takes as input $M_{\mathtt{pk}}, \mathtt{ID}_A$ and the random $r \in \mathbb{R}_{\mathtt{ID\text{-}KEM}}(M_{\mathtt{pk}})$, and outputs a key $K \in \mathbb{K}_{\mathtt{ID\text{-}KEM}}(M_{\mathtt{pk}})$ and the encapsulation of the key $C \in \mathbb{C}_{\mathtt{ID\text{-}KEM}}(M_{\mathtt{pk}})$.

  $$(K, C) \leftarrow \mathbb{E}_{\mathtt{ID\text{-}KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A;\ r)$$

  We may also use the interface $\mathbb{E}_{\mathtt{ID\text{-}KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A)$ by assuming $\mathbb{E}_{\mathtt{ID\text{-}KEM}}$ samples randomness r in the algorithm.
- **Decapsulate** $\mathbb{D}_{\mathtt{ID\text{-}KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A, D_A, C)$: This deterministic algorithm takes as input $M_{\mathtt{pk}}, \mathtt{ID}_A, D_A$ and C, and outputs the encapsulated key K in C or a failure symbol

$\perp$ if C is an invalid encapsulation.

$$(K \text{ or } \perp) \leftarrow \mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A, D_A, C),$$

Consider the two-stage game in Table 3.2 between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of the ID-KEM and a challenger.

Table 3.2: ID-KEM Security Formulation

ID-IND Adversarial Game
1. $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID-KEM}}(1^k)$.
2. $(s, \mathtt{ID}^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathtt{ID-KEM}}}(M_{\mathtt{pk}})$.
3. $(K_0, C^*) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}^*)$.
4. $K_1 \leftarrow \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}})$.
5. $b \leftarrow \{0, 1\}$.
6. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathtt{ID-KEM}}}(M_{\mathtt{pk}}, C^*, s, \mathtt{ID}^*, K_b)$.

In the games s is some state information and $\mathcal{O}_{\mathtt{ID-KEM}}$ denotes oracles to which the adversary has access. We will be interested in the CCA2 attack model where the adversary has access to two oracles:

1. A private key extraction oracle which, on input of $\mathtt{ID} \neq \mathtt{ID}^*$, will output the corresponding value of $D_{\mathtt{ID}}$.

2. A decapsulation oracle which, on input an identity ID and encapsulation of its choice, will return the encapsulated key. This is subject to the restriction that in the second phase $\mathcal{A}_2$ is not allowed to call this oracle with the pair $(C^*, \mathtt{ID}^*)$.

The adversary's advantage is defined to be

$$\mathtt{Adv}^{\mathtt{ID-IND-CCA2}}_{\mathtt{ID-KEM}}(\mathcal{A}) = \mid 2\Pr[b' = b] - 1 \mid .$$

An ID-KEM is considered to be ID-IND-CCA2 secure, if for all PPT adversaries $\mathcal{A}$, the advantage in the game above is a negligible function of the security parameter k.

In the hybrid encryption, we also need a DEM which is a symmetric-key encryption. As the DEM uses a different key derived by the KEM to encrypt each message, we only need a one-time symmetric-key encryption with proper security properties.

A one-time symmetric-key encryption consists of two deterministic polynomial-time algorithms with the key, message and ciphertext spaces defined by $\mathbb{K}_{\mathsf{SK}}(1^k)$, $\mathbb{M}_{\mathsf{SK}}(1^k)$ and $\mathbb{C}_{\mathsf{SK}}(1^k)$ given the security parameter k:

- $\mathbb{E}_{\mathsf{SK}}(K, m)$: The encryption algorithm takes a secret key $K \in \mathbb{K}_{\mathsf{SK}}(1^k)$ and a message $m \in \mathbb{M}_{\mathsf{SK}}(1^k)$ as input, and outputs the ciphertext $C \in \mathbb{C}_{\mathsf{SK}}(1^k)$.

$$C \leftarrow \mathbb{E}_{\mathsf{SK}}(K, m)$$

- $\mathbb{D}_{\mathsf{SK}}(K, C)$: Given a secret key K and a cipertext C, the algorithm outputs the plaintext m or a failure symbol $\perp$.

$$(m \text{ or } \perp) \leftarrow \mathbb{D}_{\mathsf{SK}}(K, C)$$

The two algorithms satisfy $\mathbb{D}_{\mathsf{SK}}(K, \mathbb{E}_{\mathsf{SK}}(K, m)) = m$ for $m \in \mathbb{M}_{\mathsf{SK}}(1^k)$ and $K \in \mathbb{K}_{\mathsf{SK}}(1^k)$.

We define the security of one-time encryption by the Find-Guess (FG) game in Table 3.3 between a challenger and an adversary $\mathcal{A}$ of a pair of PPT algorithms $(\mathcal{A}_1, \mathcal{A}_2)$:

In the game m_0 and m_1 are of equal length from the message space and s is some state information. $\mathcal{O}_{\mathsf{SK}}$ is the oracle that the adversary can access depending on the attack model. For our interest, the following model is required.

- CCA Model. In the model, the adversary has the access to a decryption oracle which on input a ciphertext C returns $\mathbb{D}_{\mathsf{SK}}(K, C)$ with K chosen in Step 3 in the game.

The adversary's advantage in the game above is defined to be

$$\mathtt{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}}(\mathcal{A}) = \mid 2\Pr[b' = b] - 1 \mid .$$

Table 3.3: DEM Security Formulation

FG Adversarial Game
1. $(s, (m_0, m_1)) \leftarrow \mathcal{A}_1(1^k)$.
2. $b \leftarrow \{0, 1\}$.
3. $K \leftarrow \mathbb{K}_{\mathtt{SK}}(1^k)$.
4. $C^* \leftarrow \mathbb{E}_{\mathtt{SK}}(K, m_b)$.
5. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathtt{SK}}}(C^*, s, m_0, m_1)$.

A one-time encryption is consider to be FG-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage in the game above is a negligible function of the security parameter k. The FG-CCA secure one-time encryptions are easy to get, such as the one-time pad encryption with a secure message authentication code algorithm [65].

Similar to the hybrid public-key encryption, a hybrid IBE $\mathcal{E} = (\mathbb{G}_{\mathtt{ID}}, \mathbb{X}_{\mathtt{ID}}, \mathbb{E}_{\mathtt{ID}}, \mathbb{D}_{\mathtt{ID}})$ construction consists of combining an ID-KEM $\mathcal{E}_1 = (\mathbb{G}_{\mathtt{ID-KEM}}, \mathbb{X}_{\mathtt{ID-KEM}}, \mathbb{E}_{\mathtt{ID-KEM}}, \mathbb{D}_{\mathtt{ID-KEM}})$ with a standard DEM $\mathcal{E}_2 = (\mathbb{E}_{\mathtt{SK}}, \mathbb{D}_{\mathtt{SK}})$ as described below. We assume that the key-space of the ID-KEM is the same as the key-space of the associated DEM. To generate $M_{\mathtt{pk}}$, for the hybrid IBE scheme, the algorithm $\mathbb{G}_{\mathtt{ID-KEM}}(1^k)$ is run. The key extraction for $\mathcal{E}$ is simply the key extraction of $\mathcal{E}_1$. The encryption and decryption algorithm of $\mathcal{E}$ are defined in Table 3.4.

Similar to the result of hybrid encryption in [65], Bentahar *et al.* obtained the following theorem concerning the security of hybrid IBE.

Theorem 3.5.1. *[Bentahar et al. [20]] Let $\mathcal{A}$ be a PPT ID-IND-CCA2 adversary of the IBE scheme $\mathcal{E}$ above. There exists PPT adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, whose running time is essentially that of $\mathcal{A}$, such that*

$$Adv_{ID}^{ID-IND-CCA2}(\mathcal{A}) \leq 2Adv_{ID-KEM}^{ID-IND-CCA2}(\mathcal{B}_1) + Adv_{DEM}^{FG-CCA}(\mathcal{B}_2).$$

Table 3.4: Hybrid IBE

$\mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}, m)$	$\mathbb{D}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}, D_{\mathtt{ID}}, C)$
• $(K, C_1) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID})$	• Parse C as $\langle C_1, C_2 \rangle$
• $C_2 \leftarrow \mathbb{E}_{\mathtt{SK}}(K, m)$	• $K \leftarrow \mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}, D_{\mathtt{ID}}, C_1)$
• Return $C = \langle C_1, C_2 \rangle$	• If $K = \perp$, return $\perp$
	• $m \leftarrow \mathbb{D}_{\mathtt{SK}}(K, C_2)$
	• Return m

3.5.2 SK-KEM1

Now we present the first KEM: SK-KEM1, based on the SK ID key construction. SK-KEM1 consists of following algorithms:

Setup $\mathbb{G}_{\mathtt{ID-KEM}}(1^k)$**.** On input 1^k, the algorithm works as follows:

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ of prime order p, an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$. Pick a random generator $P_2 \in \mathbb{G}_2^*$ and set $P_1 = \psi(P_2)$.

2. Pick a random $s \in \mathbb{Z}_p^*$ and compute $P_{pub} = sP_1$.

3. Pick two cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_t \rightarrow \{0,1\}^\ell$ for some integer $\ell > 0$.

4. Output the master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, \ell, P_1, P_2, P_{pub}, H_1, H_2)$ and the master secret key $M_{\mathfrak{sk}} = s$.

Extract $\mathbb{X}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$**.** Given an identifer string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathfrak{pk}}$ and $M_{\mathfrak{sk}}$, the algorithm returns $D_A = \frac{1}{s+H_1(\mathtt{ID}_A)} P_2$.

Encapsulate $\mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}_A)$**.** Given the master public key $M_{\mathfrak{pk}}$ and the recipient identity $\mathtt{ID}_A$, the algorithm works as follows:

1. Randomly choose $r \in \mathbb{Z}_p^*$.

2. Compute $Q_A = H_1(\mathtt{ID}_A)P_1 + P_{pub}$ and $g^r = \hat{e}(P_1, P_2)^r$.

3. Compute $C = rQ_A$ and $K = H_2(g^r)$.

4. Return (K, C).

Decapsulate $\mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A, D_A, C)$. Given the master public key $M_{\mathtt{pk}}$, the recipient's identity $\mathtt{ID}_A$, private key D_A and the encapsulation C, the algorithm works as follows:

1. Return $K = H_2(\hat{e}(C, D_A))$.

Similar to SK-IBE, the isomorphism ψ is not required in the implementation of the scheme.

The security of the scheme can be summarised by the following theorem.

Theorem 3.5.2. *SK-KEM1 is secure against adaptive chosen ciphertext attacks provided that the q-GBCAA1$_{1,2}$ assumption is sound, and H_1 and H_2 are random oracles.*

*Specifically, suppose there exists an ID-IND-CCA2 adversary $\mathcal{A}$ against SK-KEM1 that has advantage $\epsilon(k)$ and running time $t(k)$. Suppose also that during the attack $\mathcal{A}$ makes at most q_1 queries on H_1 and q_D **Decapsulation** queries. Then there exists an algorithm $\mathcal{B}$ to solve the $(q_1 - 1)$-GBCAA1$_{1,2}$ problem with advantage and time*

$$\begin{aligned} Adv_{\mathcal{B}}^{(q_1-1)-GBCAA1_{1,2}}(k) &\geq \tfrac{\epsilon(k)}{q_1}, \\ t_{\mathcal{B}}(k) &\leq t(k) + O((q_1 + q_D)\chi), \end{aligned}$$

where χ is time of the DBIDH algorithm.

Proof: Given an instance of the $(q_1 - 1)$-GBCAA1$_{1,2}$ problem $(xP_1,\ h_0,\ (h_1,\ \frac{1}{h_1+x}P_2),\ \ldots, (h_{q_1-1},\ \frac{1}{h_{q_1-1}+x}P_2))$ with a set of pairing parameter where $h_i \in_R \mathbb{Z}_p^*$ for $0 \leq i \leq q_1 - 1$ and the DBIDH$_{1,1}$ oracle $\mathcal{O}_{DBIDH}$, $\mathcal{B}$ simulates $\mathbb{G}_{\mathtt{ID-KEM}}$ to generate the system parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p,\ \hat{e},\ \psi,\ \ell,\ P_1,\ P_2,\ xP_1,\ H_1,\ H_2)$, i.e. using x as the master secret key, which it does not know. H_1 and H_2 are two random oracles controlled by $\mathcal{B}$.

$\mathcal{B}$ randomly chooses $1 \leq I \leq q_1$ and interacts with $\mathcal{A}$ in the following way:

- $H_1(\mathtt{ID}_i)$: $\mathcal{B}$ maintains a list of tuples $(\mathtt{ID}_i, h_i, D_i)$ as explained below. We refer to this initially empty list as H_1^{list}. When $\mathcal{A}$ queries the oracle H_1 at a point on $\mathtt{ID}_i$, $\mathcal{B}$ responds as follows:

  - If $\mathtt{ID}_i$ already appears on the H_1^{list} in a tuple $(\mathtt{ID}_i, h_i, D_i)$, then $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = h_i$.

- Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $(\mathtt{ID}_I, h_0, \perp)$ into the tuple list and responds with $H_1(\mathtt{ID}_I) = h_0$.
- Otherwise, $\mathcal{B}$ selects a random integer $h_i (i > 0)$ from the $(q_1 - 1)$-GBCAA1$_{1,2}$ instance which has not been chosen by $\mathcal{B}$ and stores $(\mathtt{ID}_i, h_i, \frac{1}{h_i + x} P_2)$ into the tuple list. $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = h_i$.

- $H_2(X_i)$: To respond to these queries, $\mathcal{B}$ maintains an initially empty list of pairs called H_2^{list}. Each entry in the list is a pair in the form (X_i, Z_i) indexed by X_i. To respond to a query on X_i, $\mathcal{B}$ does the following operations:
  - If a pair (X_i, Z_i) is on the list, then $\mathcal{B}$ responds with Z_i.
  - Otherwise, for each pair (C_j, K_j) on list $\mathcal{L}_D$ (a list maintained in the **Decapsulation** specified later), $\mathcal{B}$ queries $\mathcal{O}_{DBIDH}$ with $((h_0 + x)P_1, C_j, X_i)$. Once $\mathcal{O}_{DBIDH}$ returns 1 for one pair, $\mathcal{B}$ inserts the pair (X_i, K_j) into H_2^{list}, meanwhile removes the pair (C_j, K_j) from $\mathcal{L}_D$ and returns K_j. For other situations, $\mathcal{B}$ continues.
  - $\mathcal{B}$ randomly chooses a string $Z_i \in \{0,1\}^\ell$ and inserts a new pair (X_i, Z_i) into the list. It responds to $\mathcal{A}$ with Z_i.

- **Extraction**$(\mathtt{ID}_i)$**:** $\mathcal{B}$ looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. $\mathcal{B}$ checks the value of D_i: if $D_i \neq \perp$, then $\mathcal{B}$ responds with D_i; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).

- **Decapsulation**$(\mathtt{ID}_i, C_i)$**:** $\mathcal{B}$ maintains an initially empty list, denoted by $\mathcal{L}_D$, of pairs in the form (C_i, K_i). To respond to the query, $\mathcal{B}$ first looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, then $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. Depending on the value of D_i for $\mathtt{ID}_i$ on H_1^{list}, $\mathcal{B}$ responds differently.
  1. If $D_i \neq \perp$, then $\mathcal{B}$ first computes $g^r = \hat{e}(C_i, D_i)$, and then queries $Z_i = H_2(g^r)$. $\mathcal{B}$ responds with Z_i.
  2. Otherwise $(D_i = \perp)$, $\mathcal{B}$ takes following actions:
     (a) For every pair (X_j, Z_j) in H_2^{list}, $\mathcal{B}$ queries $\mathcal{O}_{DBIDH}$ with $((h_0+x)P_1, C_i, X_j)$. If $\mathcal{O}_{DBIDH}$ returns 1 for one query, then $\mathcal{B}$ returns the corresponding Z_j.
     (b) Otherwise, $\mathcal{B}$ randomly chooses $K \in \{0,1\}^\ell$ and inserts (C_i, K) into the list $\mathcal{L}_D$. Finally $\mathcal{B}$ returns K.

- **Challenge:** At some point $\mathcal{A}$'s first stage will terminate and it will return a challenge identity $\mathtt{ID}^*$. If $\mathcal{A}$ has not called H_1 with input $\mathtt{ID}^*$ then $\mathcal{B}$ does so for it. If the corresponding value of $D_{\mathtt{ID}^*}$ is not equal to $\perp$ then $\mathcal{B}$ will terminate (**Event 2**). $\mathcal{B}$ chooses a random value of $r \in \mathbb{Z}_p^*$ and a random value K^* in $\{0,1\}^\ell$, and returns (K^*, rP_1) as the challenge. For simplicity, if $(\mathtt{ID}^*, rP_1)$ has been queried on the **Decapsulation** query, $\mathcal{B}$ tries another random r.

- **Guess:** Once $\mathcal{A}$ outputs its guess, $\mathcal{B}$ answers the q_1-GBCAA1$_{1,2}$ challenge in the following way.

1. For each tuple (X_j, Z_j) in H_2^{list}, $\mathcal{B}$ queries $\mathcal{O}_{DBIDH}$ with $((h_0 + x)P_1, rP_1, X_j)$. If $\mathcal{O}_{DBIDH}$ returns 1, $\mathcal{B}$ outputs X_j as the answer to the challenge.
2. If no tuple found on the list, $\mathcal{B}$ failed (**Event 3**).

Claim: If algorithm $\mathcal{B}$ does not abort during the simulation, then algorithm $\mathcal{A}$'s view is identical to its view in the real attack.

Proof: $\mathcal{B}$'s responses to H_1 queries are uniformly and independently distributed in $\mathbb{Z}_p^*$ as in the real attack because of the behavior of the **Setup** phase in the simulation. H_2 is modeled as a random oracle which requires that for each unique input, there should be only one response. We note that the simulation substantially makes use of the programmability of random oracle and the access to the $\text{DBIDH}_{1,1}$ oracle to guarantee the unique response for every H_2 query. There are two subcases in the simulation.

- The adversary queries the **Decapsulation** oracle on $(\texttt{ID}^*, C_i)$. Although $\mathcal{B}$ cannot compute $K = \hat{e}(C_i, \frac{1}{h_0+x}P_2)$ (note that if the game does not abort, $D_{\texttt{ID}^*} = \frac{1}{h_0+x}P_2$), $\mathcal{B}$ makes use of $\mathcal{O}_{DBIDH}$ to test if K has been queried to H_2 as an input X_i. If K has been queried, $\mathcal{B}$ uses the existing response. Otherwise, $\mathcal{B}$ randomly generates the response and keeps a record in $\mathcal{L}_D$.

- The adversary queries on $H_2(X_i)$. If (X_i) has not been queried before on H_2, $\mathcal{B}$ should make sure that the response must be consistent with the possible existing response generated in the **Decapsulation** queries. Again, $\mathcal{B}$ exploits the access to the $\text{DBIDH}_{1,1}$ oracle. By testing $\hat{e}(C_j, \frac{1}{h_0+x}P_2) \stackrel{?}{=} X_i$ for all C_j's in $\mathcal{L}_D$ and, if the equation holds, returning the corresponding response in $\mathcal{L}_D$, $\mathcal{B}$ guarantees that the output in this query is consistent with the one in the **Decapsulation** query.

The responses in other types of query are valid as well. Hence the claim is founded.

We now evaluate the probability that $\mathcal{B}$ does not abort the game. Event 3 implies that the value $\hat{e}(C^*, \frac{1}{h_0+x}P_2)$, which is the encapsulated key in the challenge encapsulation, is not queried on H_2 in the simulation. Since H_2 is a random oracle, $\Pr[\mathcal{A} \text{ wins}|\texttt{Event 3}] = \frac{1}{2}$. We have

$$\begin{aligned}
\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins}|\texttt{Event 3}]\Pr[\texttt{Event 3}] + \Pr[\mathcal{A} \text{ wins}|\overline{\texttt{Event 3}}]\Pr[\overline{\texttt{Event 3}}] \\
&\leq \Pr[\texttt{Event 3}] + \frac{1}{2}(1 - \Pr[\texttt{Event 3}]) = \frac{1}{2} + \frac{1}{2}\Pr[\texttt{Event 3}]. \\
\Pr[\mathcal{A} \text{ wins}] &\geq \Pr[\mathcal{A} \text{ wins}|\overline{\texttt{Event 3}}]\Pr[\overline{\texttt{Event 3}}] \\
&= \frac{1}{2}(1 - \Pr[\texttt{Event 3}]) = \frac{1}{2} - \frac{1}{2}\Pr[\texttt{Event 3}].
\end{aligned}$$

So, we have $\Pr[\overline{\texttt{Event 3}}] \geq \epsilon(k)$. We note that $\overline{\texttt{Event 2}}$ implies $\overline{\texttt{Event 1}}$ because of the rules of the game. Overall, we have

$$\Pr[\mathcal{B} \text{ wins}] = \Pr[\overline{\texttt{Event 3}} \wedge \overline{\texttt{Event 2}}] \geq \frac{\epsilon(k)}{q_1}.$$

□

We note that as the proof is based on the gap assumption, in principle the reductions requires the queried encapsulation C_j to be in $\mathbb{G}_1$.

3.5.3 SK-KEM2

If one is not willing to accept the gap assumption used by SK-KEM1, here we present another KEM: SK-KEM2, which has a security reduction based on ℓ-BCAA1. We will first construct an ID-OW-CPA secure IBE, then apply a generic ID-KEM conversion to obtain SK-KEM2.

By using the SK key construction, we construct an ID-OW-CPA secure IBE referred to as Basic-SK-IBE, which is specified by the following four algorithms:

Setup $\mathbb{G}_{\mathtt{ID}}(1^k)$**.** On input 1^k, the algorithm works as follows:

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ of prime order p, an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$. Pick a random generator $P_2 \in \mathbb{G}_2^*$ and set $P_1 = \psi(P_2)$.
2. Pick a random $s \in \mathbb{Z}_p^*$ and compute $P_{pub} = sP_1$.
3. Pick two cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_t \rightarrow \{0,1\}^n$ for some integer $n > 0$.
4. Output the master public key $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, H_1, H_2)$ and the master secret key $M_{\mathtt{sk}} = s$.

The message space is $\mathbb{M} = \{0,1\}^n$, the ciphertext space is $\mathbb{C} = \mathbb{G}_1^* \times \{0,1\}^n$, and the randomness space is $\mathbb{R} = \mathbb{Z}_p$.

Extract $\mathbb{X}_{\mathtt{ID}}(M_{\mathtt{pk}}, M_{\mathtt{sk}}, \mathtt{ID}_A)$**.** Given an identifer string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathtt{pk}}$ and $M_{\mathtt{sk}}$, the algorithm returns $D_A = \frac{1}{s+H_1(\mathtt{ID}_A)} P_2$.

Encrypt $\mathbb{E}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}_A, m;\ r)$**.** Given a message $m \in \mathbb{M}$, $\mathtt{ID}_A$, $M_{\mathtt{pk}}$ and the randomness r, the following steps are performed.

1. Compute $Q_A = H_1(\mathtt{ID}_A)P_1 + P_{pub}$ and $g^r = \hat{e}(P_1, P_2)^r$.

2. Compute $U = rQ_A$ and $V = m \oplus H_2(g^r)$.

3. Set the ciphertext to $C = \langle U, V \rangle$.

Decrypt $\mathbb{D}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}_A, D_A, C)$. Given a ciphertext $C = \langle U, V \rangle \in \mathbb{C}$, $\mathtt{ID}_A$, D_A and $M_{\mathtt{pk}}$, the algorithm outputs

$$V \oplus H_2(\hat{e}(U, D_A)).$$

We have the following security result of the scheme.

Theorem 3.5.3. *Basic-SK-IBE is secure against ID-OW-CPA adversaries provided that H_1 and H_2 are random oracles and the ℓ-BCAA1 assumption is sound. Specifically, suppose that there is an algorithm $\mathcal{A}$ which breaks the scheme in terms of ID-OW-CPA with advantage $\epsilon(k)$, and let q_1 and q_2 be the number of queries that $\mathcal{A}$ makes to H_1 and H_2 respectively. Then there is an algorithm $\mathcal{B}$ essentially in time of $O(time(\mathcal{A}))$ to solve the $(q-1)$-$BCAA1_{1,2}$ problem with the advantage*

$$\boldsymbol{Adv}_{\mathcal{B}}^{(q_1-1)-BCAA1_{1,2}}(k) \approx \frac{(1 - q_2/2^n) \cdot \epsilon(k)}{q_1}.$$

Proof: Algorithm $\mathcal{B}$ is given as input a random $(q_1 - 1)$-BCAA1$_{1,2}$ instance $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p,$ $\hat{e}$, ψ, n, P_1, P_2, xP_1, h_0, $(h_1, \frac{1}{h_1+x}P_2)$, $\ldots$, $(h_i, \frac{1}{h_i+x}P_2)$, $\ldots$, $(h_{q_1-1}, \frac{1}{h_{q_1-1}+x}P_2))$ where x is a random element from $\mathbb{Z}_p^*$. Algorithm $\mathcal{B}$ finds $\hat{e}(P_1, P_2)^{1/(x+h_0)}$ by interacting with $\mathcal{A}$ as follows:

Algorithm $\mathcal{B}$ first simulates algorithm $\mathbb{G}_{\mathtt{ID}}(1^k)$ by setting $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e},$ ψ, P_1, P_2, xP_1, $H_1, H_2)$, i.e. setting $P_{pub} = xP_1$ and H_1 and H_2 are the random oracles controlled by $\mathcal{B}$. Now $\mathcal{B}$ randomly choose $1 \leq I \leq q_1$ and starts to respond to queries as follows.

- $H_1(\mathtt{ID}_i)$: $\mathcal{B}$ maintains an initially empty list of tuples $(\mathtt{ID}_i, h_i, d_i)$ indexed by $\mathtt{ID}_i$ as explained below. We refer to this list as H_1^{list}. When $\mathcal{A}$ queries the oracle H_1 at a point $\mathtt{ID}_i$, $\mathcal{B}$ responds as follows:
  - If $\mathtt{ID}_i$ already appears on the H_1^{list} in a tuple $(\mathtt{ID}_i, h_i, d_i)$, then $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = h_i$.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $(\mathtt{ID}_I, h_0, \perp)$ into the tuple list and responds with $H_1(\mathtt{ID}_I) = h_0$.
  - Otherwise, $\mathcal{B}$ selects a random integer $h_i (i > 0)$ from $(q_1 - 1)$-BCAA1 challenge which has not been chosen by $\mathcal{B}$ and stores $(\mathtt{ID}_i, h_i, \frac{1}{h_i+x}P_2)$ into the tuple list. $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = h_i$.

- $H_2(X_i)$: At any time algorithm $\mathcal{A}$ can issue queries to the random oracle H_2. $\mathcal{B}$ maintains a list H_2^{list} with tuples of the form (X_i, ζ_i) indexed by X_i. To respond to a query on X_i, $\mathcal{B}$ does the following operations:
  - If a tuple indexed by X_i is on the list, then $\mathcal{B}$ responds with ζ_i.
  - Otherwise, $\mathcal{B}$ randomly chooses a string $\zeta_i \in \{0,1\}^n$ and inserts a new tuple (X_i, ζ_i) into the list. It responds to $\mathcal{A}$ with ζ_i.
- **Extraction**($\mathtt{ID}_i$): $\mathcal{B}$ first looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, then $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. $\mathcal{B}$ then checks the value d_i: if $d_i \neq \perp$, $\mathcal{B}$ responds with d_i; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).
- **Challenge:** At some point $\mathcal{A}$'s first stage will terminate and it will return a challenge identity $\mathtt{ID}^*$. If $\mathcal{A}$ has not called H_1 with input $\mathtt{ID}^*$ then $\mathcal{B}$ does so for it. If the corresponding value of $D_{\mathtt{ID}^*}$ is not equal to $\perp$ then $\mathcal{B}$ will terminate (**Event 2**). Algorithm $\mathcal{B}$ chooses a random value of $r \in \mathbb{Z}_p^*$ and a random value V^* in $\{0,1\}^n$. It computes $U^* = rP_1$ and sets the challenge ciphertext to be

$$C^* = (U^*, V^*).$$

  Note that because $D_{\mathtt{ID}^*}$ should be $\frac{1}{h_0+x}P_2$, for a genuine challenge ciphertext we should have

$$m^* = V^* \oplus H_2(\hat{e}(U^*, D_{\mathtt{ID}^*})) = V^* \oplus H_2(\hat{e}(rP_1, \frac{1}{h_0+x}P_2)).$$

  This challenge ciphertext is now passed to algorithm $\mathcal{A}$'s second stage. Note, due to the rules of the game, $\mathcal{B}$ will not terminate unexpectedly when responding to extraction queries made once $\mathcal{A}$ has been given the challenge ciphertext.
- **Guess:** At some point algorithm $\mathcal{A}$ responds with its guess as to the value of the underlying plaintext m'. $\mathcal{B}$ does the following operation to respond to the $(q_1 - 1)$-BCAA1 challenge.
  - Compute $\zeta^* = m' \oplus V^*$.
  - Go through H_2^{list} to find tuples (X_i, ζ^*).
  - If no tuple or more than one tuple is found, fail the game.
  - Return $T = X_i^{1/r}$ as the response to the $(q_1 - 1)$-BCAA1 challenge.

By the rules of the game, $\overline{\texttt{Event 2}}$ implies $\overline{\texttt{Event 1}}$. Hence if $\mathcal{A}$ indeed chooses the I-th queried identifier on H_1 as the challenge identity, then the game won't abort. We have

$$\Pr[\mathcal{B} \text{ does not abort}] = \Pr[\overline{\texttt{Event 2}}] \geq 1/q_1.$$

Let **Event 3** be that in the game $X^* = \hat{e}(U^*, D_{\mathtt{ID}^*})$ is queried on H_2. As H_2 is a random oracle, we know that $\mathcal{A}$ has advantage at most $1/2^n$ if **Event 3** does not happen.

Again as H_2 is random oracle, in the game that there are two tuples with $X' \neq X^*$ such that $H_2(X') = H_2(X^*) = \zeta^*$ happens with probability at most $q_2/2^n$.

Overall, we have

$$\Pr[\mathcal{B} \text{ wins}] \approx \frac{(1 - q_2/2^n)\epsilon(k)}{q_1}.$$

□

To get a secure ID-KEM, we now apply a generic conversion which transforms an ID-OW-CPA secure probabilistic IBE into an ID-IND-CCA2 secure ID-KEM.

The generic conversion [20] works as follows: Let the IBE algorithm be denoted $\mathbb{E}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}, m;\ r)$ and the decryption algorithm be denoted $\mathbb{D}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}, D_{\mathtt{ID}}, C)$, where $D_{\mathtt{ID}}$ is the output from the extraction algorithm $\mathbb{X}_{\mathtt{ID}}(M_{\mathtt{pk}}, M_{\mathtt{sk}}, \mathtt{ID})$. We assume the message space of $\mathbb{E}_{\mathtt{ID}}$ is given by $\mathbb{M}_{\mathtt{ID}}(M_{\mathtt{pk}})$ and the space of randomness is given by $\mathbb{R}_{\mathtt{ID}}(M_{\mathtt{pk}})$. The construction uses two cryptographic hash functions:

$$H_3 : \mathbb{M}_{\mathtt{ID}}(M_{\mathtt{pk}}) \rightarrow \mathbb{R}_{\mathtt{ID}}(M_{\mathtt{pk}}) \text{ and } H_4 : \mathbb{M}_{\mathtt{ID}}(M_{\mathtt{pk}}) \rightarrow \{0,1\}^{\ell}$$

for $\ell \in \mathbb{Z}$: the length of the resulting keys. Using this we construct an ID-KEM as follows.

Table 3.5: A Generic ID-KEM Conversion

$\mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID})$	$\mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}, D_{\mathtt{ID}}, C)$
• $m \leftarrow \mathbb{M}_{\mathtt{ID}}(M_{\mathtt{pk}})$	• $m \leftarrow \mathbb{D}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}, D_{\mathtt{ID}}, C)$
• $r \leftarrow H_3(m)$	• If $m = \perp$, return $\perp$
• $C \leftarrow \mathbb{E}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}, m;\ r)$	• $r \leftarrow H_3(m)$
• $K \leftarrow H_4(m)$	• If $C \neq \mathbb{E}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}, m;\ r)$, return $\perp$
• Return (K, C)	• $K \leftarrow H_4(m)$
	• Return K

From [20] we have the following theorem concerning the security of the construction above.

Theorem 3.5.4. *If $\mathbb{E}_{ID}$ is an ID-OW-CPA secure ID-based encryption scheme and H_3 and H_4 are modeled as random oracles then the construction above is secure against adaptive chosen ciphertext attack. Specifically, if $\mathcal{A}$ is a PPT algorithm that breaks the ID-KEM construction above using a chosen ciphertext attack, then there exists a PPT algorithm $\mathcal{B}$, with*

$$Adv^{ID-IND-CCA2}_{ID-KEM}(\mathcal{A}) \leq 2(q_3 + q_4 + q_D) \cdot Adv^{ID-OW-CPA}_{ID}(\mathcal{B}) + \frac{2q_D\gamma(M_{pt})}{|\,\mathbb{R}_{ID}(M_{pt})\,|},$$

where q_3, q_4 and q_D are the number of queries made by $\mathcal{A}$ to H_3, H_4 and the decryption oracle respectively.

When we instantiate this generic construction with Basic-SK-IBE, we have

$$\frac{\gamma(M_{pt})}{|\,\mathbb{R}_{\mathtt{ID}}(M_{pt})\,|} \approx \frac{1}{p},$$

and the **Encapsulate** and **Decapsulate** algorithm of the full secure ID-KEM: SK-KEM2, are detailed in Table 3.6. As SK-IBE, the computation $Q_{\mathtt{ID}}$ in $\mathbb{D}_{\mathtt{ID-KEM}}$ can be pre-

Table 3.6: SK-KEM2

$\mathbb{E}_{\mathtt{ID-KEM}}(M_{pt}, \mathtt{ID}_A)$	$\mathbb{D}_{\mathtt{ID-KEM}}(M_{pt}, \mathtt{ID}_A, D_A, C)$
• $m \leftarrow \{0,1\}^n$	• Parse C as $\langle C_1, C_2 \rangle$
• $r \leftarrow H_3(m)$	• $\alpha \leftarrow \hat{e}(C_1, D_A)$
• $Q_A \leftarrow P_{pub} + H_1(\mathtt{ID}_A)P_1$	• $m = C_2 \oplus H_2(\alpha)$
• $C_1 = rQ_A$	• $r \leftarrow H_3(m)$
• $C_2 = m \oplus H_2(\hat{e}(P_1, P_2)^r)$	• $Q_A \leftarrow P_{pub} + H_1(\mathtt{ID}_A)P_1$
• $K \leftarrow H_4(m)$	• $U \leftarrow rQ_A$
• Return $(K, \langle C_1, C_2 \rangle)$	• If $C_1 \neq U$, return $\perp$
	• $K \leftarrow H_4(m)$
	• Return K

computed.

By combining Theorem 3.5.3 and Theorem 3.5.4, we have the following result regarding the security of SK-KEM2.

Theorem 3.5.5. *SK-KEM2 is secure against ID-IND-CCA2 adversaries provided that* $H_i (1 \leq i \leq 4)$ *are random oracles and the* ℓ*-BCAA1 assumption is sound. Specifically, suppose there exists an ID-IND-CCA2 adversary* $\mathcal{A}$ *against SK-KEM2 that has non-negligible advantage* $\epsilon(k)$*. Suppose also that during the attack* $\mathcal{A}$ *makes at most* q_D *decryption queries and at most* q_i *queries on* H_i *for* $1 \leq i \leq 4$ *respectively (note that* H_i *could be queried directly by* $\mathcal{A}$ *or indirectly by an extraction query, a decryption query or the challenge operation). Then there exists a PPT algorithm* $\mathcal{B}$ *to solve the* $(q_1 - 1)$*-BCAA1*$_{1,2}$ *problem with advantage*

$$\mathbf{Adv}_{\mathcal{B}}^{(q_1-1)\text{-}\mathbf{BCAA1}_{1,2}}(k) \geq \frac{1-q_2/2^n}{2q_1(q_3+q_4+q_D)}[\epsilon(k) - \frac{2q_D}{p}].$$

A few points are worth mentioning. First, by including rQ_A in H_2 we can get a more efficient reduction for both SK-KEM1 and SK-KEM2. Second, we can slightly tweak SK-KEM2 to SK-KEM2′, which then is a generalised PSEC-KEM [101] in the identity-based KEM setting, as in Table 3.7 with

$$H_2' : \mathbb{G}_1 \times \mathbb{G}_t \rightarrow \{0,1\}^n \text{ and } H_3' : \{0,1\}^n \rightarrow \mathbb{Z}_p^* \times \{0,1\}^\ell.$$

The security analysis of SK-KEM2′ essentially follows the same strategy as SK-KEM2:

Table 3.7: SK-KEM2′

$\mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A)$	$\mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}, D_A, C)$
• $m \leftarrow \{0,1\}^n$	• Parse C as $\langle C_1, C_2 \rangle$
• $r \| K \leftarrow H_3'(m)$	• $\alpha \leftarrow \hat{e}(C_1, D_A)$
• $Q_A \leftarrow P_{pub} + H_1(\mathtt{ID}_A)P_1$	• $m = C_2 \oplus H_2'(C_1, \alpha)$
• $C_1 = rQ_A$	• $r \| K \leftarrow H_3'(m)$
• $C_2 = m \oplus H_2'(C_1, \hat{e}(P_1, P_2)^r)$	• $Q_A \leftarrow P_{pub} + H_1(\mathtt{ID}_A)P_1$
• Return $(K, \langle C_1, C_2 \rangle)$	• $U \leftarrow rQ_A$
	• If $C_1 \neq U$, return $\perp$
	• Return K

First, by a very similar reduction as Theorem 5 [20], one can prove that the generic PSEC-KEM can convert an ID-OW-CPA secure ID-based encryption to an ID-IND-CCA2 secure

ID-KEM. Then, by Theorem 3.5.3, we prove the security of SK-KEM2′.

3.6 Efficiency Discussion and Comparison

Owing to the SK key construction, the SK-IBE and SK-KEMs are very efficient on computation cost and bandwidth overhead. In the following, by comparing the two schemes with other IBEs in the literature, we demonstrate that SK-IBE in Section 3.4 and SK-KEMs in Section 3.5 are the most efficient ones in terms of operation among the existing IBEs in the literature. Before giving the factual comparison details, we explain the rationale of choosing the compared schemes.

1. A chosen scheme should have a valid proof in the standard model or with resorting to random oracles.

2. As efficiency is our main focus and a scheme constructed in the standard model is less efficient than the corresponding variant of using random oracles with same parameter size, for simplicity we only give detailed comparison of those schemes with random oracles, but informatively compare those schemes secure in the standard model.

We choose the following schemes for comparison.

- **BF-IBE**. This is the Boneh-Franklin IBE [19], which uses the Fujisaki-Okamoto conversion to achieve ID-IND-CCA2 security in the random oracle model. This is a "full domain hash" scheme.

- **BB_1-IBE**. This is the first Boneh-Boyen scheme in [10] but with using hash functions on the identifier and ciphertext. The scheme was first detailed in a submission to IEEE P1363.3 [9] and achieves ID-IND-CCA2 security in the random oracle model. This is an efficient "commutative blinding" scheme.

- **Gentry-IBE**. This is an IBE scheme using an "exponent inversion"-type key construction [82]. The scheme has a tight security reduction in the standard model but based on a stronger assumption ℓ-DABDHE [82] than ℓ-BDHI.

- **SK-IBE**. This is the scheme presented in Section 3.4, which is also an "exponent inversion" scheme.

- **BF-KEM**. This is the KEM adaptation of BF-IBE. The construction is also applying the Bentahar *et al.*'s generic transformation on the basic ID-IND-CPA secure BF-IBE (BasicIdent) [19] as SK-KEM does in Section 3.5. We note that the trivial KEM construction from the basic BF-IBE like ECIES-KEM [101] will achieve higher efficiency but with a security proof in the random oracle model based on the stronger GBDH assumption than BDH [117].

- **BB_1-KEM**. This is the KEM adaptation of BB_1-IBE first shown in [9] with security proof in the random oracle model.

- **SK-KEM2**. This is the ID-KEM in Section 3.5.3. It's easy to find out that SK-KEM1 has shorter ciphertext and is slightly faster than SK-KEM2.

There are a few IBEs and ID-KEMs with security proofs in the standard model. Most of those IBEs including Waters-IBE [162], CS-IBE [66] and Naccache-IBE [131] use Waters' ID key construction or its variant. The last two are the improvements of Waters' scheme. All of these "commutative blinding" IBEs are less efficient than BB_1-IBE from the same category. Boneh and Boyen presented a second IBE, called BB_2-IBE in [10], using an "exponent inversion"-type key construction. They formally proved that the scheme is a sID-IND-CPA secure IBE in the standard model. To achieve sID-IND-CCA2 security [56] in the standard model, one has to apply some other inefficient transformation such as a zero-knowledge construction. In [9], a BB_2-IBE variant with random oracles was briefly mentioned but no

scheme details and security analysis was given[1]. BMW-KEM [39] and Kiltz-KEM [108] are two ID-KEMs constructed using the variant of Waters' key construction with security proofs in the standard model. However, none of them are of better efficiency than BB_1-KEM from the same category.

Below we provide the extensive comparison data regarding the chosen schemes' computation efficiency of the Extract, Encrypt/Encapulate and Decrypt/Decapsulate algorithm, and the communication bandwidth including the size of the system parameters, the master secret key, the private key and the ciphertext/capsulation.

We choose the 80-bit security level with the Type-1 pairing and the 128-bit security level with the Type-3 pairing as the comparison bed. We choose the 80-bit and 128-bit security level because the first is commonly adopted currently and the latter is believed to be necessary in the future and the Type-1 (resp. Type-3) pairings are generally faster than the other three types on the 80-bit (resp. 128-bit) security level. Although there is no efficient isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ for Type-3 pairings, this is not an obstacle to implement the chosen schemes, though the Gentry-IBE has to be slightly modified by choosing an extra generator in $\mathbb{G}_2$.

Although formal security reductions are of great importance to validate the security of a cryptographic scheme, as discussed in Section 2.3, a loose reduction cannot suggest meaningful parameters for a fixed security level in practice. Because all the chosen schemes to be compared have a loose security reduction except the Gentry's scheme which has a tight reduction but based on a stronger assumption, we follow the recommendation from NIST for comparing ECC and RSA key sizes [132] to choose the system parameters. The same approach is also adopted in the IEEE P1363.3 submissions [9, 14].

[1] By applying the Fujisaki-Okamoto conversion like BF-IBE and SK-IBE or adopting the MAC-tag construction like BB_1-IBE, the enhance scheme is secure in the random oracle model based on the ℓ-BDHI assumption. However, the new scheme is about 50% (resp. 20%) slower in encryption with Type-1 (resp. Type-3) pairings, and is about 40% slower with Type-1 pairings and, if full optimization is unavailable, 20% slower with Type-3 pairings in decryption than SK-IBE. The scheme has larger system parameter, private key and ciphertext than SK-IBE.

As in [9, 52, 14], we will compare the schemes in terms of computation efficiency by reference to the cost of a multiplication in $\mathbb{G}_1$. We assume pairing-friendly curves [28] are used and the point compression technique is applied.

Table 3.8: Relative Cost of Operations and Bandwidth in Pairing-Based Schemes

	Type 1[1]	Type 3[2]
Security Level	80	128
Size of Elements		
$\in \mathbb{G}_1$	512 bits	256 bits
$\in \mathbb{G}_2$	512 bits	512 bits
$\in \mathbb{G}_t$	1024 bits	3072 bits
Relative Cost of Operation		
Mul in $\mathbb{G}_1$	1	1
Fix-base Mul in $\mathbb{G}_1$[3]	0.2	0.2
Mul in $\mathbb{G}_2$	1	3
Fix-base Mul in $\mathbb{G}_2$[3]	0.2	0.6
Exp in $\mathbb{G}_t$	0.2	3
Fix-base Exp in $\mathbb{G}_t$[3]	0.05	0.6
Pairing	2.1/1[4]	20
Double-Pairing	3.7/1.6[4]	28[5]
Hash in $\mathbb{G}_1$	2.2	free
Hash in $\mathbb{G}_2$	2.2	3

1. We assume the Tate pairing with the embedding degree 2 is used.
2. We assume the Tate pairing with the embedding degree 12 is used.
3. We assume that the fix-base multiplication or exponentiation uses 20% time of a general multiplication or exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_t$ respectively.
4. The pre-computation is used to speed-up pairings.
5. We use the result of optimising pairing product in [90].

The computation performance is compared in Table 3.9 and the communication bandwidth is compared in Table 3.10. Both comparisons show that SK-IBE and SK-KEM2 are the most efficient ones in terms of operation and bandwidth requirement. The SK schemes are significantly faster in Encrypt algorithm than the BF schemes. When the cost ratio of pairing and multiplication in $\mathbb{G}_1$ is small, for example on the chosen Type-1 pairing group,

Table 3.9: Relative Computation Efficiency of IBEs from Pairings

	BF-IBE	BB_1-IBE	Gentry-IBE	SK-IBE	BF-KEM	BB_1-KEM	SK-KEM2
Hard Problem	BDH	BDH	ℓ-DABDHE	ℓ-BDHI	BDH	$BDH^{(3)}$	ℓ-BDHI
Private Key Generation							
Hash to $\mathbb{G}_2$	1				1		
Mul in $\mathbb{G}_2$	1	$\overline{2}^{(1)}$	$\overline{6}$	$\overline{1}$	1	$\overline{2}$	$\overline{1}$
Type-1							
Relative Cost	3.2	2	6	1	3.2	2	1
Optimised Cost$^{(2)}$	3.2	0.4	1.2	0.2	3.2	0.4	0.2
Type-3							
Relative Cost	6	6	6	3	6	6	3
Optimised Cost$^{(2)}$	6	1.2	3.6	0.6	6	1.2	0.6
Encrypt/ Encapsulate							
Hash to $\mathbb{G}_2$	1				1		
Mul in $\mathbb{G}_1$	$1+\overline{1}$	$\overline{3}$	$\overline{2}$	$\overline{2}$	$1+\overline{1}$	$\overline{3}$	$\overline{2}$
Exp in $\mathbb{G}_t$		$\overline{1}$	$\overline{4}$	$\overline{1}$		$\overline{1}$	$\overline{1}$
Pairings	1				1		
Type-1							
Relative Cost	6.3	3.2	2.8	2.2	6.3	3.2	2.2
Optimised Cost$^{(2)}$	4.4	0.65	0.6	0.45	4.4	0.65	0.45
Type-3							
Relative Cost	25	6	14	5	25	6	5
Optimised Cost$^{(2)}$	24.2	1.2	2.8	1.0	24.2	1.2	1.0
Decrypt/ Decapsulate							
Mul in $\mathbb{G}_1$	$\overline{1}$	$\overline{1}$		$\overline{1}$	$\overline{1}$		$\overline{1}$
Mul in $\mathbb{G}_2$			$\overline{1}$				
Exp in $\mathbb{G}_t$		$\overline{1}$	2				
Pairings	1	2	2	1	1	2	1
Type-1							
Relative Cost	3.1	4.9	5.6	3.1	3.1	3.7	3.1
Optimised Cost$^{(2)}$	1.2	1.85	4.8	1.2	1.2	1.6	1.2
Type-3							
Relative Cost	21	$31^{(4)}$	49	21	21	28	21
Optimised Cost$^{(2)}$	20.2	$28.6^{(4)}$	46.2	20.2	20.2	28	20.2

1. Symbols $\overline{m}$ and n denote m fix-based multiplications or exponentiations and n general operations respectively.
2. The cost has considered the optimisation of fix-based multiplication or exponentiation.
3. BB_1-KEM was claimed in [9] to be secure based on the BDH assumption. However, no proof is given. On the other hand, one can certainly prove the security of the scheme based on the GBDH assumption by adopting the technique in [10]. The same technique is used in Theorem 4.6.1 in Section 4.6.2.
4. It appears with Type-3 pairings, BB_1-IBE performs slightly better by checking the validity of the ciphertext with $(c_1, c_0) \stackrel{?}{=} (g_3^s g_1^{H(\mathtt{ID})s}, g^s)$ [9].

Table 3.10: Communication Bandwidth of IBEs from Pairings

	BF-IBE	BB_1-IBE	Gentry-IBE	SK-IBE	BF-KEM	BB_1-KEM	SK-KEM2
System Parameters							
# elts. in $\mathbb{G}_1$	2	3	2	2	2	3	2
# elts. in $\mathbb{G}_2$	0	0	$3+1^{(1)}$	0	0	0	0
# elts. in $\mathbb{G}_t$	0	1	0	0	0	1	0
Param. Size							
Type-1	1024	2560	2560	1024	1024	2560	1024
Type-3	512	3840	2560	512	512	3840	512
Master Secret							
# elts. in $\mathbb{G}_1$	0	1	0	0	0	1	0
# elts. in $\mathbb{Z}_p$	1	3	1	1	1	3	1
Secret Size							
Type-1	160	992	160	160	160	992	160
Type-3	256	1024	256	256	256	1024	256
Private Key							
# elts. in $\mathbb{G}_2$	1	2	3	1	1	2	1
# elts. in $\mathbb{Z}_p$	0	0	3	0	0	0	0
Key Size							
Type-1	512	1024	2016	512	512	1024	512
Type-3	512	1024	2304	512	512	1024	512
Cipher Text Fixed-len. Part							
# elts. in $\mathbb{G}_1$	1	2	1	1			
# elts. in $\mathbb{G}_t$	0	0	3	0			
# elts. in $\mathbb{Z}_p$	0	1	0	0			
Ciphertext Size							
Type-1	$512+r^{(2)}+n^{(3)}$	$1184^{(4)}$	$3584^{(4)}$	$512+r+n$			
Type-3	$256+r+n$	$1024^{(4)}$	$9472^{(4)}$	$256+r+n$			
KEM Capsule Fixed-len. Part							
# elts. in $\mathbb{G}_1$					1	2	1
# elts. in $\mathbb{Z}_p$					0	0	0
Capsule Size							
Type-1					$512+r$	1024	$512+r$
Type-3					$256+r$	512	$256+r$

1. In Gentry-IBE with the Type-3 pairings, an extra generator $g_2 \in \mathbb{G}_2$ should be chosen and $h_{\mathtt{ID},i} = (h_i g_2^{-r_{\mathtt{ID},i}})^{1/(\alpha-\mathtt{ID})}$ with $h_i \in \mathbb{G}_2$ for $i = 1, 2, 3$ [82].
2. r is the size of the used randomness.
3. n is the size of the message.
4. The message must be securely embedded in $\mathbb{G}_t$ and so cannot be longer than the size of $\mathbb{G}_t$.

the BB_1 schemes, the Gentry scheme and SK schemes have similar performance when operations are fully optimised and the SK schemes are the fastest. While, when the ratio is large, for example on the chosen Type-3 pairing group, SK-IBE and SK-KEM2 are much faster than the BB_1 schemes and Gentry-IBE in decryption. SK-IBE and SK-KEM2 have the same bandwidth efficiency as the BF scheme, while is more economical than the BB_1 schemes. Gentry-IBE has very large ciphertext size.

3.7 On Cheon's Attack

As observed by Moni Naor that an ID key construction is in fact a signature scheme, the result of the Extract algorithm is a signature D_A on the message $\mathtt{ID}_A$ signed under the master secret key. For the security of an IBE, the basic requirement is that the adversary itself should not be able to generate a private key associated with an identifier, i.e. the signature scheme should be existentially unforgeable under the chosen-message attack [91]. The SK ID key construction has been proven to be a secure signature in the sense of existential unforgeability provided that the ℓ-SDH assumption is sound in $\mathbb{G}_2$ [169].

As discussed in Section 2.6, the ℓ-SDH problem is not as hard as the DH problem. Recently Cheon presented an algorithm showing that the computational complexity of ℓ-DHI (and ℓ-SDH) can be reduced by $O(\sqrt{\ell})$ from that of the discrete logarithm problem [41]. Here we restate Cheon's two main results:

Theorem 3.7.1. *Let P be an element of prime order p in an Abelian group. Suppose that ℓ is a positive divisor of $p-1$. If $P, P_1 := \alpha P$ and $P_\ell := \alpha^\ell P$ are given, α can be computed in $O(\log p \cdot (\sqrt{(p-1)/\ell} + \sqrt{\ell}))$ group operations using $O(\max\{\sqrt{(p-1)/\ell}, \sqrt{\ell}\})$ memory.*

Theorem 3.7.2. *Let P be an element of prime order p in an Abelian group. Suppose that ℓ is a positive divisor of $p+1$. If $P_i := \alpha^i P$ for $i = 1, 2, \ldots, 2\ell$ are given, α can be computed in $O(\log p \cdot (\sqrt{(p+1)/\ell} + \ell))$ group operations using $O(\max\{\sqrt{(p+1)/\ell}, 2\ell\})$ memory.*

Now we investigate the possible impact of Cheon's algorithm on the implementation of SK-IBE and SK-KEMs. We will look at this issue from two aspects, one regarding the

direct threat to the system in practice and another regarding the implication on the security proof in theory.

For the real threat, we have to assume that the adversary can obtain private keys because the adversary can be internal parties of a system and may also compromise other parties' private key. By assuming a *single* adversary $\mathcal{A}$ has gathered ℓ pairs of public/private keys $(h_i, \frac{1}{s+h_i}P_2)$ with $h_i = H_1(\mathtt{ID}_i)$ and h_i is different from each other, by using Cheon's algorithm $\mathcal{A}$ can try to recover the master secret key in the following manner.

- Randomly sample $h_0 \in \mathbb{Z}_p^*$ different from h_i for $1 \leq i \leq \ell$.
- Set $\alpha = s + h_0$ which $\mathcal{A}$ does not know, and

$$Q = \frac{1}{(s+h_1)\cdots(s+h_\ell)}P_2.$$

- For $j = 0, \ldots, \ell - 1$, $\mathcal{A}$ computes

$$\alpha^j Q = \frac{(s+h_0)^j}{(s+h_1)\cdots(s+h_\ell)}P_2 = \sum_{i=1}^{\ell} \frac{c_{ij}}{s+h_i}P_2$$

  where $c_{ij} \in \mathbb{Z}_p$ are computable from h_i's.

- Given $\alpha^j Q$ for $0 \leq j \leq \ell - 1$, use Cheon's algorithm to compute α and so $s = \alpha - h_0$.

By Theorem 3.7.2 the complexity of the above attack to recover the master secret key s in SK-IBE/KEMs is with $O(\log p \cdot (\sqrt{2(p+1)/\ell} + \ell/2))$ group operations using $O(\max\{\sqrt{2(p+1)/\ell}, \ell\})$ memory. Note that if there is a divisor d of $p-1$ with $d \leq \ell$ and $d \approx \ell$, then the attack is still workable with similar computational complexity by Theorem 3.7.1.

The value ℓ in the above attack is restricted by two factors in practice: the compromised private keys by a single adversary and the divisors of $p+1$ or $p-1$ for the chosen prime order p of $\mathbb{G}_1$. Our observation of this attack is that it does not pose a serious threat to a

practical deployment of SK-IBE/KEMs. First, we note that in a secure IBE system, each user has to be authenticated by the KGC before extracting his private key corresponding to its identifier. Hence, for a practical system with a proper safeguard, it should be impossible for an adversary to gather a very large number of private keys and the number of the compromised keys should be upper-bounded by the number of users in the system. Normally, for an enterprise-scale system, the number of users hardly exceeds 2^{30}. Second, for a system allowing ℓ private keys to be compromised at the same time requiring strict $2k$-bit DL equivalent security level, i.e. at the k-bit security level by assuming DL, DH and BDH have the same computational complexity in the chosen groups, one can choose a larger group to compensate the complexity deduction due to the attack. By the Boneh-Boyen's lower bound on ℓ-SDH, $2k + \log_2 \ell$-bit size group is enough for security compensation in general if $\ell \leq \sqrt[3]{2^{2k}}$. Regarding the concrete security parameter $k \geq 80$, if $\ell \leq 2^{30}$, $2k + 30$-bit p is sufficient to maintain the k-bit security against the Cheon's attack in general. In practice, we can properly choose pairing parameters such that the system can have better performance at the same time allow more keys to be compromised.

For a SK-IBE/KEMs system requiring the 80-bit security at the same time allowing up to 2^{54} private keys to be compromised against the Cheon's attack, one can use following parameters with a supersingular curve such as

$$y^2 = x^3 - 1 \text{ or } y^2 = x^3 + x$$

- **Example 1**. Supersingular curve defined over $\mathbb{F}_q$ with the embedding degree 2 where

q is a 512-bit prime.

$$
\begin{aligned}
q &= 995860612994766932199268928277920353719694142738581766557094\backslash \\
&\quad 5086069853643996895360939651304657594151071524576002028677176\backslash \\
&\quad 0747702108866165359149943053220 3 \\
p &= 2^{180} + 2^{11} - 1 \\
p-1 &= 2 \cdot 9643 \cdot 800113 \cdot (\text{a 54 bit prime}) \cdot (\text{a 93 bit prime}) \\
p+1 &= 2^{11} \cdot 3 \cdot 2731 \cdot (\text{a 63 bit prime}) \cdot (\text{a 94 bit prime})
\end{aligned}
$$

An implementation of SK-IBE/KEMs in Example 1 uses groups whose order is only 20-bit larger than the commonly chosen group order (160-bit) for other schemes based on the BDH assumption. A SK-IBE/KEM in Example 1 is about 10% slower than the implementation with 160-bit groups of curves defined over a 512-bit base field, and the bandwidth is only 20 bits larger (SK-KEM1 retains the same ciphertext size). On the other hand, even using this larger parameter, SK-IBE/KEMs is still slightly faster than BB_1-IBE/KEM and much faster than BF-IBE in encryption and Gentry-IBE in decryption when those scheme are implemented with 160-bit groups of the same curve defined over a same size field. Moreover, if a single adversary cannot compromise more than 2^{20} keys, which is very likely to happen in practical systems, by the Boneh-Boyen's lower bound on ℓ-SDH, SK-IBE/KEMs in Example 1 could guarantee a higher security level than BB_1-IBE/KEM.

Similarly for higher security level, proper parameter can also be selected, for example the following one in Example 2, with which SK-IBE/KEMs can maintain the 128-bit security level against the Cheon's attack, when $\ell < 2^{54}$.

- **Example 2**. Barreto-Naehrig (BN) curve $y^2 = x^3 + b$ over $\mathbb{F}_q$ with the embedding

degree 12 at the 143-bit security level (p is a 286-bit prime).

$$\begin{aligned} q &= 699360801201262467352579831411041896836011888211558644343 7\backslash \\ &\quad 55234069413355165306441508 03 \\ b &= 200160724934413405860459930896586352947669852102919861179 3\backslash \\ &\quad 87306153334869946182466364 32 \\ p &= 699360801201262467352579831411041896836011804583764149853 9\backslash \\ &\quad 18344365475072931235678633 09 \\ p-1 &= 2^2 \cdot 3 \cdot 11 \cdot 17 \cdot 571 \cdot 9001 \cdot (\text{a 54 bit prime}) \cdot (\text{a 199 bit prime}) \\ p+1 &= 2 \cdot 5 \cdot 7 \cdot (\text{a 29 bit prime}) \cdot (\text{a 109 bit prime}) \cdot (\text{a 142 bit prime}) \end{aligned}$$

We remark that the Cheon's algorithm requires nontrivial amount of storage, which may become another hurdle to cross for the algorithm's application in the distant future. This is more obvious at high security levels such as the 112-bit or 128-bit security. For example, consider the following curve parameters:

- **Example 3**. BN curve curve with the embedding degree 12 at the 112-bit security level (p is a 224-bit prime) from [28].

$$\begin{aligned} p &= 26959946667149205758383469736921690242718878200571531029749 23\backslash \\ &\quad 5996909 \\ p-1 &= 2^2 \cdot 3 \cdot (\text{a 22 bit prime}) \cdot (\text{a 25 bit prime}) \cdot (\text{a 27 bit prime}) \cdot \\ &\quad (\text{a 38 bit prime}) \cdot (\text{a 73 bit prime}) \\ p+1 &= 2 \cdot 5 \cdot 7^2 \cdot 29 \cdot 59 \cdot 3361 \cdot 6491 \cdot 44449 \cdot (\text{a 34 bit prime}) \cdot \\ &\quad (\text{a 42 bit prime}) \cdot (\text{a 90 bit prime}) \end{aligned}$$

- **Example 4**. BN curve with the embedding degree 12 at the 128-bit security level (p is a 256-bit prime) from [28].

$$\begin{aligned} p &= 11579208923731493687268856124447174205803559598884026858 44\backslash \\ &\quad 88757999429535617037 \\ p-1 &= 2^2 \cdot 3 \cdot 7^2 \cdot 189239 \cdot (\text{a 25 bit prime}) \cdot (\text{a 33 bit prime}) \cdot \\ &\quad (\text{a 36 bit prime}) \cdot (\text{a 137 bit prime}) \\ p+1 &= 2 \cdot 38113 \cdot 39563 \cdot (\text{a 98 bit prime}) \cdot (\text{a 127 bit prime}) \end{aligned}$$

In Example 3, if $\ell \approx 2^{42}$, the memory required by the algorithm will be at least 2^{96} bytes. For less memory cost, ℓ has to be bigger, which requires more keys to be compromised and the next targeted ℓ has to be 2^{73}. In Example 4, the algorithm requires 2^{116} bytes storage if $\ell \approx 2^{36}$ or at least 2^{105} bytes storage if $\ell \geq 2^{98}$.

Now we look at the impact of the attack on the security proof. As explained in Section 2.3, we take the asymptotic view on the reductions in this thesis. SK-IBE/KEMs are asymptotically secure by assuming ℓ-BDHI is hard. ℓ is upper bounded by the number of compromised private keys, which is a constant number in practice. An interesting point of the reductions in Section 3.4 and 3.5 is that the concrete complexity of the underlying assumption ℓ-BDHI is changing with the bound ℓ. Moreover, such ℓ is linked with the query number on H_1. In the reductions, for a fixed p, the more queries are issued on H_1 (so the greater ℓ), the less complexity the underlying problem has. This problem arises because for simplicity the reductions does not differentiate the direct queries on H_1 from those indirect H_1 queries trigged by the Extract query, which will really affect the bound ℓ in practice. However, in practice the adversary cannot threaten the security of the master secret key by merely executing the hash function H_1. This demonstrates that part of a reduction in the random oracle model does not necessarily coincide with the corresponding attacking behavior in practice.

3.8 Conclusion

In this chapter, we first constructed a complete IBE (SK-IBE) using the Sakai-Kasahara's identity key construction and formally analyse the scheme's security in the Boneh-Franklin IBE model. To encrypt arbitrary long messages, we then proposed two ID-KEMs (SK-KEM1 and SK-KEM2) with the same key construction. Through the detailed comparison, we demonstrated that these schemes offer the best performance among the existing IBEs. Finally we investigated the impact of Cheon's algorithm of ℓ-DHI on the proposed schemes

and concluded that the three schemes can maintain the required security level and at the same time work with high performance by using proper pairing parameters in practice.

Chapter 4

Efficient Certificateless Public Key Encryption

In last chapter, we have presented a number of practical IBEs. These IBEs following Shamir's proposal have the inherent key escrow property. As the key escrow property may not be desirable in some environments, it is natural to ask whether a secure cryptographic system without the key escrow property and at the same time as IBE without certificates is constructible. In 2003, Al-Riyami and Paterson introduced a new public key paradigm called certificateless public key cryptography (CL-PKC) which is exactly such type of system. After the certificateless public key encryption (CL-PKE) was introduced, there have been a number of constructions. Some of them are not very efficient and some others are insecure against certain attacks. In this chapter, we first revisit various CL-PKE security models and define a strong security notion for this type of encryption. Then by making a simple observation on some IBEs and PKEs, we propose a general approach to build efficient CL-PKEs from IBEs and following this approach we construct three efficient concrete CL-PKE schemes and formally analyse their security in the defined strong security model.

4.1 Introduction

To address the threat of the impersonation attack on PKC, a common strategy is to introduce into the system an authority trusted by all users. With the interventions of the authority, the impersonation attack launched by a malicious user can be thwarted by different methods.

One method is that the authority explicitly provides a guarantee that one user's ownership of a claimed public key is authentic. The certificate-based public key cryptography takes this approach. Each user obtains from the authority a certificate which securely binds the user identity with the user's public key by a signature generated by the authority. By this approach, an infrastructure to issue certificates has to be constructed and also one has to verify certificates to obtain others' authentic public keys. Such infrastructure can be very complicated and faces many challenges in practice, such as the efficiency and scalability of the infrastructure.

The second method is that users use their identity directly as their public keys and so the public key authenticity problem is trivial and certificates are no longer necessary. Each user's private key has to be generated by the authority. This is the approach taken by the IBC. However, in this type of system, the authority knows every user's private key, i.e. the system has the inherent key-escrow function, and no method can prevent a curious authority from decrypting users' ciphertexts or impersonating a user.

Though the IBC paradigm offers great advantage of simplicity over the certificate-based PKC, the key escrow property is not desirable in some settings. The natural question arises that whether a public key system as IBC certificate-free and at the same time as PKC key-escrow-free is constructible. In 2003, Al-Riyami and Paterson brought forth the notion of "Certificateless Public Key Cryptography" (CL-PKC) [5] to respond to this challenge. In the CL-PKC, a user has a public key generated by himself and his private key is determined

by two pieces of secret information: one secret associated with the user's identity is passed by the authority and the other associated with the public key is generated by the user himself. Moreover, one secret is not computable from the other, so the authority cannot compute the private key corresponding to a user's public key. Hence the CL-PKC is key-escrow-free.

The approach against the impersonation attack in the CL-PKC is not to prove authenticity of a public key by a certificate. Instead, a CL-PKC guarantees that even if a malicious user successfully replaces a victim's public key with its own choice and so could know the secret associated with the public key but not the other secret obtained by the victim user from the authority, it still cannot generate a valid signature or decrypt the message encrypted under the false public key and the victim's identifier. This will certainly reduce the interest of launching the impersonation attack.

Since the CL-PKE was introduced, there have been a number of general and concrete constructions. As intended by CL-PKE to combine the advantage of both IBE and PKE, most of these schemes integrate an IBE with a PKE in some ways. However some were later found to be insecure and some others are not very efficient. In this chapter, we make a heuristic observation on some constructions of IBE and PKE, and based on the observation we propose a general approach to constructing CL-PKEs and develop three efficient concrete CL-PKEs and formally analyse their security.

4.2 CL-PKE Model

Here we first specify the CL-PKE algorithms and then revisit various CL-PKE security models and define a strong security notion for this type of encryption.

For a CL-PKE scheme we define the public key, message, ciphertext and randomness spaces by $\mathbb{P}_{\mathsf{CL}}(\cdot)$, $\mathbb{M}_{\mathsf{CL}}(\cdot)$, $\mathbb{C}_{\mathsf{CL}}(\cdot)$ and $\mathbb{R}_{\mathsf{CL}}(\cdot)$. These spaces are parametrised by the master public key $M_{\mathfrak{pk}}$, and hence by the security parameter k. A CL-PKE scheme consists of following algorithms:

- **CL.Gen**(1^k). On input 1^k, the probabilistic algorithm generates the master secret key $M_{\mathtt{sk}}$ and the master public key $M_{\mathtt{pk}}$.

$$(M_{\mathtt{sk}}, M_{\mathtt{pk}}) \leftarrow \mathbf{CL.Gen}(1^k).$$

- **CL.PartialKey**$(M_{\mathtt{sk}}, M_{\mathtt{pk}}, \mathtt{ID}_A)$. The probabilistic algorithm takes $M_{\mathtt{sk}}$, $M_{\mathtt{pk}}$ and an arbitrary identity string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A as input and returns a partial private key D_A corresponding to $\mathtt{ID}_A$.

$$D_A \leftarrow \mathbf{CL.PartialKey}(M_{\mathtt{sk}}, M_{\mathtt{pk}}, \mathtt{ID}_A).$$

- **CL.SecretVal**$(M_{\mathtt{pk}}, \mathtt{ID}_A)$. The probabilistic algorithm takes $M_{\mathtt{pk}}$ and the identity string $\mathtt{ID}_A$ as input and returns the secret value X_A associated with the entity A.

$$X_A \leftarrow \mathbf{CL.SecretVal}(M_{\mathtt{pk}}, \mathtt{ID}_A).$$

- **CL.PrivateKey**$(M_{\mathtt{pk}}, D_A, X_A)$. The deterministic algorithm takes $M_{\mathtt{pk}}$, D_A and X_A as input and outputs the private key S_A of entity A.

$$S_A \leftarrow \mathbf{CL.PrivateKey}(M_{\mathtt{pk}}, D_A, X_A).$$

- **CL.PublicKey**$(M_{\mathtt{pk}}, X_A, \mathtt{ID}_A)$. The deterministic algorithm takes $M_{\mathtt{pk}}$ and X_A as input and outputs the public key P_A of the entity A.

$$P_A \leftarrow \mathbf{CL.PublicKey}(M_{\mathtt{pk}}, X_A, \mathtt{ID}_A).$$

- **CL.Encrypt**$(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, m;\ r)$. The algorithm takes $M_{\mathtt{pk}}$, $\mathtt{ID}_A$, P_A, a message $m \in \mathbb{M}_{\mathtt{CL}}(M_{\mathtt{pk}})$ and the randomness $r \in \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}})$ as input and returns the ciphertext $C \in \mathbb{C}_{\mathtt{CL}}(M_{\mathtt{pk}})$ of the message m. We also use the interface **CL.Encrypt** $(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, m)$ by assuming that r is sampled in the algorithm when the context is clear.

$$C \leftarrow \mathbf{CL.Encrypt}(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, m;\ r).$$

- **CL.Decrypt**$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. The deterministic algorithm takes $M_{\mathfrak{pk}}$, $\mathtt{ID}_A$, S_A and a ciphertext C as input, and outputs the value of the plaintext m or a failure symbol $\perp$.

$$\{m \text{ or } \perp\} \leftarrow \textbf{CL.Decrypt}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C).$$

Similar to IBE and PKE, to cope with probabilistic ciphers, we will require that not too many choices for r encrypt a given message to a given ciphertext. To formalize this concept we let $\gamma(M_{\mathfrak{pk}})$ be the least upper bound such that

$$\mid \{r \in \mathbb{R}_{\mathtt{CL}}(M_{\mathfrak{pk}}) : \mathbb{E}_{\mathtt{CL}}(M_{\mathfrak{pk}}, \mathtt{ID}, P_{\mathtt{ID}}, m;\ r) = C\} \mid \leq \gamma(M_{\mathfrak{pk}})$$

for every $\mathtt{ID}$, $P_{\mathtt{ID}} \in \mathbb{P}_{\mathtt{CL}}(M_{\mathfrak{pk}})$, $m \in \mathbb{M}_{\mathtt{CL}}(M_{\mathfrak{pk}})$ and $C \in \mathbb{C}_{\mathtt{CL}}(M_{\mathfrak{pk}})$. We say a CL-PKE is γ-**uniform** if $\gamma(M_{\mathfrak{pk}}) / \mid \mathbb{R}_{\mathtt{CL}}(M_{\mathfrak{pk}}) \mid < \gamma$.

Following Al-Riyami-Paterson's CL-PKE security formulation, we can define various security notions for this type of encryption. These security notions are defined by two games. Game 1 is conducted between a challenger and a Type-I adversary $\mathcal{A}_I$ of two PPT algorithms $(\mathcal{A}_{I_1}, \mathcal{A}_{I_2})$. A Type-I adversary does not know the master secret key and can replace an entity's public key with its choice. This type of adversary simulates those who may impersonate a party by providing others with a false public key. Game 2 is conducted between a challenger and a Type-II adversary $\mathcal{A}_{II}$ of two PPT algorithms $(\mathcal{A}_{II_1}, \mathcal{A}_{II_2})$. A Type-II adversary knows the master secret key (so every entity's partial private key). This type of adversary simulates a malicious KGC adversary which eavesdrops the communications between two of its subscribers or may publish false public keys. And a CL-PKE against Type-II adversaries also naturally achieves the forward secrecy of the master secret key, i.e. the compromise of the master secret key would not reveal those messages encrypted under authentic public keys before the leakage of the master secret key.

In the games above, s is some state information and $\mathcal{O}_{\mathtt{CL}}$ are the oracles that the

Table 4.1: CL-IND-PKE Games

Game 1: Type-I Adversarial	Game 2: Type-II Adversarial
1. $(M_{pt}, M_{st}) \leftarrow$ **CL.Gen**(1^k).	1. $(M_{pt}, M_{st}) \leftarrow$ **CL.Gen**(1^k).
2. $(s, \mathrm{ID}^*, m_0, m_1) \leftarrow \mathcal{A}_{I-1}^{\mathcal{O}_{\mathrm{CL}}}(M_{pt})$.	2. $(s, \mathrm{ID}^*, m_0, m_1) \leftarrow \mathcal{A}_{II-1}^{\mathcal{O}_{\mathrm{CL}}}(M_{pt}, M_{st})$.
3. $b \leftarrow \{0,1\}, r \leftarrow \mathbb{R}_{\mathrm{CL}}(M_{pt})$.	3. $b \leftarrow \{0,1\}, r \leftarrow \mathbb{R}_{\mathrm{CL}}(M_{pt})$.
4. $C^* \leftarrow$ **CL.Encrypt**$(M_{pt}, \mathrm{ID}^*, P_{\mathrm{ID}^*}, m_b;\ r)$.	4. $C^* \leftarrow$ **CL.Encrypt**$(M_{pt}, \mathrm{ID}^*, P_{\mathrm{ID}^*}, m_b;\ r)$.
5. $b' \leftarrow \mathcal{A}_{I-2}^{\mathcal{O}_{\mathrm{CL}}}(s, M_{pt}, C^*, \mathrm{ID}^*, P_{\mathrm{ID}^*}, m_0, m_1)$.	5. $b' \leftarrow \mathcal{A}_{II-2}^{\mathcal{O}_{\mathrm{CL}}}(s, M_{pt}, M_{st}, C^*, \mathrm{ID}^*, P_{\mathrm{ID}^*}, m_0, m_1)$.

Table 4.2: CL-OW-PKE Games

Game 1: Type-I Adversarial	Game 2: Type-II Adversarial
1. $(M_{pt}, M_{st}) \leftarrow$ **CL.Gen**(1^k).	1. $(M_{pt}, M_{st}) \leftarrow$ **CL.Gen**(1^k).
2. $(s, \mathrm{ID}^*) \leftarrow \mathcal{A}_{I-1}^{\mathcal{O}_{\mathrm{CL}}}(M_{pt})$.	2. $(s, \mathrm{ID}^*) \leftarrow \mathcal{A}_{II-1}^{\mathcal{O}_{\mathrm{CL}}}(M_{pt}, M_{st})$.
3. $m \leftarrow \mathbb{M}_{\mathrm{CL}}(M_{pt}), r \leftarrow \mathbb{R}_{\mathrm{CL}}(M_{pt})$.	3. $m \leftarrow \mathbb{M}_{\mathrm{CL}}(M_{pt}), r \leftarrow \mathbb{R}_{\mathrm{CL}}(M_{pt})$.
4. $C^* \leftarrow$ **CL.Encrypt**$(M_{pt}, \mathrm{ID}^*, P_{\mathrm{ID}^*}, m;\ r)$.	4. $C^* \leftarrow$ **CL.Encrypt**$(M_{pt}, \mathrm{ID}^*, P_{\mathrm{ID}^*}, m;\ r)$.
5. $m' \leftarrow \mathcal{A}_{I-2}^{\mathcal{O}_{\mathrm{CL}}}(s, M_{pt}, C^*, \mathrm{ID}^*, P_{\mathrm{ID}^*})$.	5. $m' \leftarrow \mathcal{A}_{II-2}^{\mathcal{O}_{\mathrm{CL}}}(s, M_{pt}, M_{st}, C^*, \mathrm{ID}^*, P_{\mathrm{ID}^*})$.

adversary can access during the game. Depending on the security model, these oracles may include the following:

- a public key broadcast oracle *Public-Key-Broadcast*, which takes as input an identifier and returns the associated public key. If necessary, the oracle will execute the **CL.PublicKey** algorithm first.

- a partial key exposure oracle *Partial-Private-Key-Extract*, which returns the partial private key associated with an identity. If necessary, the oracle will execute the **CL.PartialKey** algorithm first. This oracle is only useful to Type-I adversaries, as a Type-II adversary can compute every partial private key using the master secret key.

- a secret value exposure oracle *Secret-Value-Extract*, which reveals the secret value of entity whose public key was not replaced. If necessary, the algorithm will execute

algorithm **CL.SecretVal** first.

- a public key replace oracle *Public-Key-Replace*, which takes as input an identifier and a public key from the public key space and replaces the current public key associated with the identifier with the provided key. This oracle is designed to simulate the situation that a party (the sender of a message) will not verify another party (the receiver of the message)'s ownership of a claimed public key and an adversary may replace anyone's public key with a false one in CL-PKE.

- a strong decryption oracle $Decrypt^S$, which takes as input a ciphertext and an identifier and outputs the decryption of the ciphertext using the the *current* private key associated with the identifier. Note that in the games, the adversary may have replaced the public key associated with an identity, and this decryption oracle is required to output the correct decryption, which can be a failure symbol as well, using the private key corresponding to the current public key even if it may not know the corresponding secret value. As this oracle may not reflect practice, a normal decryption oracle is defined as follows:

- a decryption oracle $Decrypt^P$, which takes as input a ciphertext and an identifier and outputs the decryption of the ciphertext using the the original (before any *Public-Key-Replace* query) private key associated with the identifier. Though this query reflects the common practice that given a ciphertext a party uses its own private key to decrypt it, some attacks of conceptional interest are not simulated (see the attack on the Al-Riyami-Paterson CL-PKE [6] in Section 4.5.4). Another conceptional decryption oracle shown below is used in some formulations.

- a decryption oracle $Decrypt^C$, which takes as input a ciphertext, an identifier, a public key and the secret value corresponding to the given public key and outputs the

decryption of the ciphertext using the private key determined by the partial key corresponding to the identifier and the given secret value in the query. If the secret value is not given in the query, then the oracle works as $Decrypt^P$.

For each type of adversary, we define two attack models in which the adversary is allowed to access different oracles.

For Type-I adversaries, we define the following two attack models.

- CCA2 model. In this model, the adversary is allowed to access the *Public-Key-Broadcast*, *Partial-Private-Key-Extract*, *Secret-Value-Extract*, *Public-Key-Replace* and $Decrypt^S$ oracles. However, there are a few restrictions on the adversary.
  - If *Public-Key-Replace* has been issued on an identity, then *Secret-Value-Extract* on the identity is disallowed.
  - *Partial-Private-Key-Extract* on $\mathtt{ID}^*$ is disallowed.
  - $Decrypt^S$ on $(\mathtt{ID}^*, C^*)$ is not allowed in $\mathcal{A}_{I-2}$ when $\mathtt{ID}^*$'s current public key $P_{\mathtt{ID}^*}$ is the same as when the challenge query is issued.
- CPA model. In this model, the adversary has the access to the similar oracles as in the CCA2 model, but the $Decrypt^S$ oracle is disallowed.

As the strong decryption oracle $Decrypt^S$ may not reflect the practice, we can define two weaker CCA2 models Type-I^P and Type-I^C in which the adversary is allowed to access the $Decrypt^P$ and $Decrypt^C$ oracle instead of $Decrypt^S$ respectively.

For Type-II adversaries we define the following two attack models.

- CCA2 model. In this model, the adversary is allowed to access the *Public-Key-Broadcast*, *Secret-Value-Extract*, *Public-Key-Replace* and $Decrypt^S$ oracles. Again, there are a few restrictions on the adversary.

  - If *Public-Key-Replace* has been issued on an identity, then *Secret-Value-Extract* on the identity is disallowed.
  - *Public-Key-Replace* on $\mathtt{ID}^*$ is disallowed.
  - *Secret-Value-Extract* on $\mathtt{ID}^*$ is disallowed.
  - $Decrypt^S$ on $(\mathtt{ID}^*, C^*)$ is not allowed in $\mathcal{A}_{II-2}$.

- CPA model. In this model, the adversary has the access to the similar oracles as in the CCA2 model, but the $Decrypt^S$ oracle is disallowed.

Similarly, we can define a weaker CCA2 model Type-II^P in which the adversary is allowed to access the $Decrypt^P$ oracle instead. There is no interest of defining Type-II^C security as it requires the adversary to know both the partial key (because of the knowledge of the master secret key) and the secret value which implies the adversary can decrypt the ciphertext on its own.

If we let MOD denote the mode of attack, either CPA or CCA2, the adversary's advantage in the indistinguishability-based game is defined to be

$$\mathtt{Adv}_{\mathtt{CL}}^{\mathtt{CL-IND-MOD}}(\mathcal{A}) = \mid 2\Pr[b' = b] - 1 \mid,$$

while, the advantage in the one-way game is given by

$$\mathtt{Adv}_{\mathtt{CL}}^{\mathtt{CL-OW-MOD}}(\mathcal{A}) = \Pr[m' = m].$$

A CL-PKE algorithm is considered to be secure, in the sense of a given goal and attack model (CL-IND-CCA2 for example) if, for any PPT Type-I (and Type-II) adversary, the advantage in the relevant game is a negligible function of the security parameter k.

There are two main differences between the model defined above and the Al-Riyami-Paterson's CL-PKE security model [5, 6]. First in the Al-Riyami-Paterson model a private key exposure oracle which returns the private key of an entity is used, while, here we provide

the adversary with the secret value exposure oracle instead. As in CL-PKE each entity has two pieces of secret information, it is natural to provide the adversary with an exposure oracle for each secret. Because the entity private key is determined by two secrets: the partial private key and the secret value, the *Secret-Value-Extract* oracle with the *Partial-Key-Extract* oracle certainly can simulate the private key exposure oracle. On the other hand, in the Al-Riyami-Paterson model given a private key and the corresponding partial key the adversary may not be able to recover the related secret value if the **CL.PrivateKey** algorithm is a one-way function such as the algorithm used in [5]. We think it is useful to design a model, which considers the possibility of leaking the secret value, for several reasons. First, the implementation of the **CL.SecretVal** or **CL.PrivateKey** algorithm may not be secure enough and the secret value may be compromised in the process. Second, it is possible that the partial private key or the secret value is stored after the private key has been generated for some reasons. The partial private key maybe is stored for degeneration to an IBE if the key can be used as an identity-based private key, and the leak of it and the private key may result in the compromise of the secret value. The secret value may be stored also for flexibility. If one wants to keep his public key unchanged but with an updated identifier, which corresponding to a new partial private key, from his previous identifier, then he needs the current secret value to compute the new private key. The *Secret-Value-Extract* oracle provides the adversary the capability to compromise the secret value.

The second difference is that in Game 2 the adversary can access the *Public-Key-Replace* oracle and the strong decryption oracle $Decrypt^S$ (this formulation has been adopted in a number of other works [118, 68, 3]), while, in the Al-Riyami-Paterson's model the *Public-Key-Replace* oracle is disallowed and the decryption oracle $Decrypt^P$ is used instead. A trivial Type-II attack applicable in the enhanced model on the Al-Riyami-Paterson CL-PKE [6] (see Section 4.5.4) shows that the new formulation defines a stronger model of

Type-II adversaries in theory.

On the relation of different security formulations, the Type-I (resp. Type-II) security is certainly stronger than the Type-I^P and Type-I^C (resp. Type-II^P) security. Because of the behavior of $Decrypt^C$, the Type-I^C security is at least as strong as the Type-I^P one. The Type-I^C attack on the Al-Riyami-Paterson PKE [6] shows that the Type-I^C security is indeed stronger than Type-I^P.

Recently, Au *et al.* [3] made an interesting observation that a malicious KGC may generate the master public/secret key pair in a special way to help it decrypt some user's ciphertext. And they presented such an attack against the Al-Riyami-Paterson CL-PKE [5]. The attack shows that it is meaningful to let a Type-II adversary generate the master keys in Game 2. The enhanced Type-II adversary of three PPT algorithm $(\mathcal{A}_{II-1}, \mathcal{A}_{II-2}, \mathcal{A}_{II-3})$ launches the following Game 2′ with a challenger, where s', s are some state information.

Table 4.3: CL-IND-PKE Game 2′

Type-II^+ Adversarial
1. $(s, M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathcal{A}_{II-1}(1^k)$.
2. $(s', \mathtt{ID}^*, m_0, m_1) \leftarrow \mathcal{A}_{II-2}^{\mathcal{O}_{\mathtt{CL}}}(s, M_{\mathtt{pk}}, M_{\mathtt{sk}})$.
3. $b \leftarrow \{0,1\}, r \leftarrow \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}})$.
4. $C^* \leftarrow \mathbf{CL.Encrypt}(M_{\mathtt{pk}}, \mathtt{ID}^*, P_{\mathtt{ID}^*}, m_b;\ r)$.
5. $b' \leftarrow \mathcal{A}_{II-3}^{\mathcal{O}_{\mathtt{CL}}}(s', M_{\mathtt{pk}}, M_{\mathtt{sk}}, C^*, \mathtt{ID}^*, P_{\mathtt{ID}^*}, m_0, m_1)$.

However, it remains an open question that a secure CL-PKE against both Type-I and Type-II^+ adversaries is constructible in the standard model. We also note that in Game 2′ the adversary generates the master public key, so a question arises that who controls any random oracle included in the master public key in the random oracle model. In [3], a

generic CL-PKE scheme is claimed to be secure in this strong model. However, no proof is given and it is unclear who (the challenger or the adversary) controls the random oracles. As the random oracle model implies that the adversary cannot make use of any special structure of a hash function in the attack. This suggests that the adversary should choose "standard" cryptographic hash functions or should demonstrate that a chosen hash function has no hidden trapdoors in practice.

On the other hand, Au *et al.*'s attack can be defeated by requiring the KGC to demonstrate the randomness of the choice of parameters. In particular, the attack against the Al-Riyami-Paterson CL-PKE [5] requires the adversary to choose from the used group a specific generator P, which supposes to be random. For its innocence the KGC can show a witness of the randomness of the generator such as a public string S with $P = H(S)$ for a cryptographic hash function H. One can refer to IEEE P1363 and ANSI X9.62 standards for examples of methods used to generate verifiably random parameters.

In this chapter, we adopt the enhanced Al-Riyami-Paterson formulation (the Type-I+Type-II security model) and conduct the security analysis of the proposed schemes in the random oracle model. However, the security reductions with random oracles cannot guarantee that when the schemes are instantiated in practice with hash functions in place of random oracles, the same security strength can be maintained. In fact it looks unwise to suggest such strong security of the presented schemes for any practical purpose.

4.3 Some Related Work

Since the CL-PKE notion was introduced, there have been a number of generic or concrete constructions. In [2], Al-Riyami presented three general constructions of CL-PKE. They are the sequential or parallel composition of a secure IBE with a secure PKE to encrypt a message m. Yum and Lee proposed yet another generic construction but by double-encryption with two secure IBEs [165].

CL-Gen1: $\mathbf{IBE.Encrypt}(M_{\mathtt{pk}}, \mathtt{ID}_A, \mathbf{PKE.Encrypt}(P_A, m))$

CL-Gen2: $\mathbf{PKE.Encrypt}(P_A, \mathbf{IBE.Encrypt}(M_{\mathtt{pk}}, \mathtt{ID}_A, m))$

CL-Gen3: $\langle \mathbf{PKE.Encrypt}(P_A, m'), \mathbf{IBE.Encrypt}(M_{\mathtt{pk}}, \mathtt{ID}_A, m \oplus m') \rangle$

CL-Gen4: $\mathbf{IBE.Encrypt}(M_{\mathtt{pk}}, \mathtt{ID}_A, \mathbf{IBE.Encrypt}(M'_{\mathtt{pk}}, \mathtt{ID}_A, m))$

In CL-Gen3, m' is a random message from the message space. To decrypt the message m, PKE.Decrypt and IBE.Decrypt are applied on the two parts of the ciphertext respectively to recover m' and $m \oplus m'$, and so finally m is recovered. In CL-Gen4, the user generates the master public/secret pair $(M'_{\mathtt{pk}}, M'_{\mathtt{sk}})$ and $M'_{\mathtt{pk}}$ is treated as the user's public key.

Unfortunately, none of those generic constructions is secure in the sense of the CL-IND-CCA2 security. CL-Gen1 is insecure against a Type-II^P adversary which given the challenge ciphertext C^* on message m_b generates C'=$\mathbf{IBE.Encrypt}(M_{\mathtt{pk}}, \mathtt{ID}^*, \mathbf{IBE.Decrypt}(M_{\mathtt{pk}}, \mathtt{ID}^*, D_{\mathtt{ID}^*}, C^*))$ and queries the $Decrypt^P$ oracle on $(\mathtt{ID}^*, C')$ to recover the challenge message m_b. Note that the Type-II^P adversary can compute the partial key $D_{\mathtt{ID}^*}$ of $\mathtt{ID}^*$ and $C' \neq C^*$ because IBE.Encrypt is a probabilistic algorithm. Similarly, CL-Gen4 is insecure against Type-II^P adversaries and CL-Gen2 is vulnerable to a Type-I^P adversary. Both Type-I^P and Type-II^P adversaries can break CL-Gen3. It appears a simple solution that links the message with the message recipient's identifier and public key and the randomness used in both encryptions can rescue the above constructions [118]. In [20], Bentahar *et al.* extended the key encapsulation mechanism (KEM) to the CL-PKE setting and proposed a generic construction from an IBE and a PKE. There have been a couple of general constructions in the standard model as well [70, 99]. All of these general constructions are not very efficient regarding both computation and communication cost. They all need double-encryption with either two IBEs or one IBE with a PKE, and the ciphertext of these schemes are longer than the used IBE or PKE's.

In [118], Libert and Quisquater showed that with slight modification the second Fujisaki-Okamoto conversion [74] can convert a CL-IND-CPA secure CL-PKE to a CL-IND-CCA2

secure scheme. In the next section we will demonstrate a similar result on the first Fujisaki-Okamoto conversion [73].

On the concrete construction, in [5] Al-Riyami and Paterson presented a CL-PKE, referred to as AP-CL-PKE1, based on a stronger assumption than BDH. The CL.Encrypt algorithm of the scheme requires three pairing operations which are very costly. Moreover, the scheme is vulnerable to the malicious KGC's attack [3], which in turn requires a verifiable random parameter generation process. In [6], an improved scheme referred to as AP-CL-PKE2 was constructed by integrating the BF-IBE with the ElGamal encryption enhanced by the Fujisaki-Okamotom conversion. However, as pointed out in [168], the scheme is insecure against Type-I^C adversaries. A trivial Type-II attack can break the scheme as well (see Section 4.5.4)[1]. In [150, 118], two constructions based on the SK-IBE [46] were presented. These schemes rely on a stronger ℓ-BDHI assumption than BDH.

Several other works are worth mentioning. In [29], the authors proposed a CL-PKE-like scheme without using pairing, which is more efficient than those schemes based on pairings. However, the scheme requires a user to execute CL.PartialKey first before generating the public key. This makes it more like a self-certified public key scheme [76] instead of CL-PKE. Moreover, the scheme unlike other CL-PKE proposals based on IBEs cannot work compatibly with exiting IBEs, i.e. the scheme cannot degenerate smoothly to an IBE. Gentry proposed a closely related security notion "Certificate-Based Encryption" (CBE) and a concrete CBE scheme [81]. The CBE security notion shares very much the similarity with CL-PKE. Al-Riyami and Paterson had attempted to construct CBE from CL-PKE by giving a specific conversion [6]. Recently, Liu *et al.* introduced a new notion "self-generated-certificate" PKE [104], which intends to address the so-called Denial-of-Decryption (DoD) attack. The authors considered that the adversary may replace an entity's public key with

[1] In 2005, we independently presented a CL-PKE [46], which shares similarity with AP-CL-PKE2. But the scheme is immune to the attacks against AP-CL-PKE2. A slight variant of the scheme is to be presented in Section 4.5.

a false one to prevent the entity from decrypting messages encrypted under the false public key. However, the threat of the DoD attack is arguable and their proposal does not solve the problem either. In particular, their solution requires an entity to publish a public encryption key with a signature generated with its certificateless private signature key. The sender first verifies the recipient's certificateless signature on the claimed public encryption key. Only if the signature is valid, the public key is used to encrypt messages to the recipient. But, the authors did not consider that the adversary may use an expired or revoked public encryption key, which has a publicly known valid signature, to launch the DoD attack. To address this issue, a public key may have to be accompanied by an expiry time and a revocation list of revoked public keys has to be maintained securely. However, it appears that with this extra management, the system will end up very much like a PKI.

4.4 Heuristic Approach to CL-PKE

Now we explain our heuristic approach to constructing CL-PKE. This approach is based on a simple observation of some existing IBEs and PKEs.

The encryption schemes based on the DH assumption or its variants essentially take the same general approach as the ElGamal encryption [72] as follows:

$$\text{PKE ciphtext}=\langle \text{DH token(s)}, \text{Hiding(message; DH value)} \rangle$$

The encrypter generates one or more DH tokens and uses the DH token(s) and the decrypter's public key to compute the DH value. Then the DH value is used as the secret to hide messages in a message hiding algorithm. The decrypter uses its private key and the DH token(s) in the ciphertext to compute the DH value and so to recover the conveyed message.

As shown in Chapter 3, most of the existing IBE schemes [19, 48, 9] make use of pairing and base their security on the BDH assumption or its variants which are the descendants of

the DH assumption. This explains that those IBE encryptions essentially adopts the same approach as those PKEs based on the DH assumption:

$$\text{IBE ciphtext}=\langle\text{pairing-DH token(s)}, \text{Hiding(message; pairing-DH value)}\rangle$$

The encrypter first computes one or more tokens which we call the pairing-DH tokens. Then the encrypter computes a value (we call it the pairing-DH value) which can be computed through pairings by the decrypter with its private key, the pairing-DH token(s) and possibly the system parameters as well. The pairing-DH value is used as the secret to hide the message by the encrypter in a message hiding algorithm and to recover the conveyed message by the decrypter.

We can see that pairing-based IBEs and DH-based PKEs have a common structure and some message hiding algorithms such as the data encapsulation mechanism and the Fujisaki-Okamoto transformation can be used in both the PKE [65, 73, 74] and the IBE [20, 114, 167]. Based on the above observation, it seems natural to use a hash function to integrate a secure IBE scheme with a secure PKE scheme to achieve a secure CL-PKE scheme, which can be presented as follows:

$$\text{CL-PKE ciphtext}= \langle\text{PKE.DH-tokens(s)}, \text{IBE.pairing-DH token(s)},$$
$$\text{Hiding}'(\text{message}; H(\text{DH value, pairing-DH value}))\rangle$$

where H is a hash function. When the PKE.DH-token(s) and IBE.pairing-DH token(s) have tokens that can be generated in the same way, then one token generated with the same randomness can be used instead. The message hiding algorithm might need to be slightly modified as the input secret value is no longer a DH or pairing-DH value but a hash value. And if those DH or pairing-DH values are only used in the hash functions in the Hiding algorithm then H is unnecessary.

An intuitive view on the security of the above construction as a CL-PKE[2] is that to

[2]The idea of using hash function to integrate an IBE with a PKE to construct a CL-PKE was first proposed and demonstrated in work [46].

recover the message, one has to obtain the hash on the DH value and the pairing-DH value which in turn requires one to know both values. While, a Type-I adversary cannot compute the pairing-DH value if the underlying IBE is secure and a Type-II adversary cannot compute the DH value if the underlying PKE is secure. Similarly, one can construct CL-KEM with the same approach.

We mention that this approach is only based on the heuristic observation. A CL-PKE or CL-KEM constructed with this approach might not be necessarily secure in the model defined in Section 4.2. On the other hand, following this approach that using hash function on the DH and the pairing-DH values to tightly integrate an IBE with a PKE, we indeed are able to construct highly efficient and secure CL-PKEs. In the following part, we will present three concrete CL-PKEs constructed in this way.

4.5 CL-PKE1

In this section, we present a CL-PKE which using a hash function integrates the BF-IBE with the ElGamal-like PKE enhanced with the Fujisaki-Okamoto conversion [73]. A slightly simpler version of the scheme, which strictly follows the general approach in Section 4.4, was first shown in an early draft of the work [46]. Here for the ease of the security analysis, we will adopt a generic construction which introduces minor extra computation overhead.

4.5.1 A Generic Conversion

Fujisaki and Okamoto [73] proposed a generic conversion to transform a weak PKE to a strongly secure PKE. Here we demonstrate that a slightly modified conversion can also be applied in the CL-PKE setting to convert a CL-OW-CPA secure CL-PKE to a CL-IND-CCA2 secure CL-PKE.

Let Π be a CL-PKE scheme with the encryption algorithm $\mathcal{E}$ and the decryption algorithm $\mathcal{D}$. Define a CL-PKE scheme $\overline{\Pi}$ with the encryption algorithm $\overline{\mathcal{E}}$ as

$$\langle C_1, C_2 \rangle \leftarrow \overline{\mathcal{E}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, m;\ \sigma)$$

where

$$\langle C_1, C_2 \rangle = \langle \mathcal{E}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, \sigma;\ G_1(m, \sigma, \mathtt{ID}_A, P_A)), m \oplus G_2(\sigma) \rangle$$

and the decryption algorithm $\overline{\mathcal{D}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, \langle C_1, C_2 \rangle)$ as

- $\sigma \leftarrow \mathcal{D}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C_1)$
- $m = C_2 \oplus G_2(\sigma)$
- If $C_1 = \mathcal{E}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, \sigma;\ G_1(m, \sigma, \mathtt{ID}_A, P_A))$, output m; otherwise output $\perp$.

and other algorithms essentially the same as Π.

$\overline{\Pi}$ shares the same parameters with Π except $\overline{M_{\mathfrak{pk}}}$, the master public system parameters of $\overline{\Pi}$ which includes Π's master public system parameters $M_{\mathfrak{pk}}$ and two extra hash functions G_1 and G_2 defined as follows:

$$G_1 : \mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathfrak{pk}}}) \times \mathbb{M}_{\Pi}(M_{\mathfrak{pk}}) \times \{0,1\}^* \times \mathbb{P}_{\overline{\Pi}}(\overline{M_{\mathfrak{pk}}}) \rightarrow \mathbb{R}_{\Pi}(M_{\mathfrak{pk}})$$
$$G_2 : \mathbb{M}_{\Pi}(M_{\mathfrak{pk}}) \rightarrow \mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathfrak{pk}}})$$

Theorem 4.5.1. *Suppose Π is a γ-uniform CL-PKE secure scheme against CL-OW-CPA attacks, then $\overline{\Pi}$ is a CL-IND-CCA2 scheme. More specifically, suppose that a Type-I (resp. Type-II) CL-IND-CCA2 adversary $\mathcal{A}$ has advantage $\epsilon(k)$ against $\overline{\Pi}$ with running time $t(k)$, making q_D decryption queries and $q_{G_1} < 2^{|\mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathfrak{pk}}})|}$ and q_{G_2} random oracle queries on G_1 and G_2 respectively. Then there exists a Type-I (resp. Type-II) CL-OW-CPA adversary $\mathcal{B}$ with advantage*

$$Adv_{\mathcal{B}}(k) \geq \frac{\epsilon(k)}{q_{G_1} + q_{G_2}}(1-\gamma)^{q_{G_1} q_D}$$

over Π in running time

$$t_{\mathcal{B}}(k) \leq t(k) + O(q_{G_1} t_{\mathcal{E}}),$$

where $t_{\mathcal{E}}$ is the cost of $\mathcal{E}$.

Proof: We show how to make use of Type-I (resp. Type-II) CL-IND-CCA2 adversary $\mathcal{A}$ against $\overline{\Pi}$ to construct a Type-I (resp. Type-II) CL-OW-CPA adversary $\mathcal{B}$ again Π. The challenger $\mathcal{T}$ starts a Type-I (resp. Type-II) CL-OW-CPA game by passing $\mathcal{B}$ the master public key $M_{\mathtt{pk}}$ and providing with the oracle access including the possible random oracles, *Public-Key-Broadcast*, *Public-Key-Replace*, *Secret-Value-Extract* and for Type-I $\mathcal{B}$ with *Partial-Private-Key-Extract* as well. In the Type-II game, $\mathcal{T}$ also gives $M_{\mathtt{sk}}$ to $\mathcal{B}$.

$\mathcal{B}$ forwards $M_{\mathtt{pk}}$ with G_1 and G_2 as the master public key $\overline{M_{\mathtt{pk}}}$ of $\overline{\Pi}$ to $\mathcal{A}$ where G_1 and G_2 are two random oracles controlled by $\mathcal{B}$. In the Type-II game, $\mathcal{B}$ also passes $M_{\mathtt{sk}}$ to $\mathcal{A}$. $\mathcal{B}$ provides $\mathcal{A}$ with the oracle access as follows, and for simplicity of presentation we assume that $\mathcal{A}$ will abide by the rules defined in the models in Section 4.2.

- $\mathcal{B}$ forwards to its CL-OW-CPA challenger $\mathcal{T}$ the queries on oracles including *Public-Key-Broadcast*, *Secret-Value-Extract*, *Public-Key-Replace*, and for Type-I $\mathcal{T}$ *Partial-Private-Key-Extract* as well, and relays the answers from the challenger to $\mathcal{A}$.

- $\mathcal{B}$ forwards to its challenger $\mathcal{T}$ the queries on the possible random oracles provided by the challenger and relays the answers to $\mathcal{A}$.

- $G_1(m_i, \sigma_i, \mathtt{ID}_i, P_i)$**:** To respond to these queries $\mathcal{B}$ maintains a list G_1^{list}. Each entry in the list is a tuple of the form $(m_i, \sigma_i, \mathtt{ID}_i, P_i, r_i, C_1^i, C_2^i)$ indexed by $(m_i, \sigma_i, \mathtt{ID}_i, P_i)$. To respond to a query, $\mathcal{B}$ does the following operations:

  - If on G_1^{list} a tuple indexed by $(m_i, \sigma_i, \mathtt{ID}_i, P_i)$ exists, then $\mathcal{B}$ responds with the corresponding r_i.
  - Otherwise,
    * $\mathcal{B}$ randomly chooses a string $r_i \in \mathbb{R}_{\Pi}(M_{\mathtt{pk}})$.
    * $\mathcal{B}$ computes $C_1^i = \mathcal{E}(M_{\mathtt{pk}}, \mathtt{ID}_i, P_i, \sigma_i;\ r_i)$ and $C_2^i = \mathbb{G}_2(\sigma_i) \oplus m_i$.
    * $\mathcal{B}$ inserts a new tuple $(m_i, \sigma_i, \mathtt{ID}_i, P_i, r_i, C_1^i, C_2^i)$ into the list and responds to $\mathcal{A}$ with r_i.

- $G_2(\sigma_i)$**:** To respond to these queries $\mathcal{B}$ maintains a list G_2^{list}. Each entry in the list is a tuple of the form (σ_i, h_i) indexed by σ_i. To respond to a query, $\mathcal{B}$ does the following operations:

  - If on the list there is a tuple indexed by σ_i, then $\mathcal{B}$ responds with the corresponding h_i.
  - Otherwise, $\mathcal{B}$ randomly chooses a string $h_i \in \mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})$ and inserts a new tuple (σ_i, h_i) into the list. It responds to $\mathcal{A}$ with h_i.

- **Decrypt**$^S(\mathtt{ID}_i, C_i)$**:** $\mathcal{B}$ takes the following steps to respond to the query:

  - $\mathcal{B}$ queries its challenger the current public key P_i associated with $\mathtt{ID}_i$ by issuing *Public-Key-Broadcast*$(\mathtt{ID}_i)$.
  - $\mathcal{B}$ parses C_i as $\langle C_1^i, C_2^i \rangle$ and searches G_1^{list} to find tuples $(*, *, \mathtt{ID}_i, P_i, *, C_1^i, C_2^i)$.

- If no such a tuple is found, then $\mathcal{B}$ outputs $\perp$.
- If more than one tuple is found, then $\mathcal{B}$ outputs $\perp$.
- $\mathcal{B}$ outputs m_i in the only found tuple.

- **Challenge:** Once $\mathcal{A}$ decides that Phase 1 is over, it outputs identity $\mathtt{ID}^*$ and two messages m_0, m_1 on which it wishes to be challenged.
  - $\mathcal{B}$ queries $\mathcal{T}$ with the *Public-Key-Broadcast*($\mathtt{ID}^*$) to get the current public key P^* associated with $\mathtt{ID}^*$.
  - $\mathcal{B}$ forwards $\mathtt{ID}^*$ as the challenge ID to $\mathcal{T}$ and gets the challenge ciphertext as C_1^*.
  - $\mathcal{B}$ randomly samples $C_2^* \in \mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})$ and replies $\mathcal{A}$ with $C^* = \langle C_1^*, C_2^* \rangle$ as the challenge ciphertext in the CL-IND-CCA2 game.

- **Guess:** Once $\mathcal{A}$ outputs its guess b'. $\mathcal{B}$ randomly chooses a σ from G_1^{list} or G_2^{list} and outputs σ as the answer of the CL-OW-CPA game.

Now we analyse $\mathcal{B}$'s probability of outputting the correct response σ to $\mathcal{T}$. We define two events.

- **Event 1** is that in the game $\mathcal{A}$ queries $G_1(*, \mathcal{D}(M_{\mathtt{pk}}, \mathtt{ID}^*, P^*, S^*, C_1^*), \mathtt{ID}^*, P^*)$ or $G_2(\mathcal{D}(M_{\mathtt{pk}}, \mathtt{ID}^*, P^*, S^*, C_1^*))$ where S^* is the private key associated with $\mathtt{ID}^*$ when the challenge is issued.
- **Event 2** is that in the game $\mathcal{A}$ differentiates $\mathcal{B}$ from a real world before Event 1 happens.

Now we look at the possibility that $C^* = \langle C_1^*, C_2^* \rangle$ is a valid ciphertext of m_b for $b \in \{0, 1\}$. For C^* to be a valid challenge ciphertext, it is required that

$$\mathcal{D}(M_{\mathtt{pk}}, \mathtt{ID}^*, P^*, S^*, C_1^*) = \sigma'$$

$$G_2(\sigma') \oplus C_2^* = m_b$$

and

$$\mathcal{E}(M_{\mathtt{pk}}, \mathtt{ID}^*, P^*, \sigma';\ G_1(m_b, \sigma', \mathtt{ID}^*, P^*)) = C_1^*.$$

As C_2^* is randomly sampled from $\mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})$, C_2^* is valid for m_0 or m_1 with equal probability. And as σ' is randomly sampled by $\mathcal{T}$ and G_1 is a random oracle, C_1^* is valid for m_0 or m_1 with equal probability as well. Hence, given C^*, $\mathcal{A}$ either finds that C^* is not a valid ciphertext for either m_0 or m_1, or (C^* is a valid ciphertext for either m_0 or m_1 with equal probability) to win the game, outputs b if C^* is a valid ciphertext for m_b.

Here we conceptually force the adversary $\mathcal{A}$ to immediately output a random $b' \in \{0, 1\}$ if it finds that C^* is an invalid challenge ciphertext. This change does not affect $\mathcal{A}$'s chance of winning the game. As G_1 and G_2 are random oracles,

$$\Pr[\mathcal{A} \text{ win} \mid \overline{\text{Event 1}}] = 1/2.$$

Then we have that

$$\begin{aligned}
\Pr[\mathcal{A}\text{ wins}] &= \Pr[\mathcal{A}\text{ wins}|\texttt{Event 1}]\Pr[\texttt{Event 1}] + \Pr[\mathcal{A}\text{ wins}|\overline{\texttt{Event 1}}]\Pr[\overline{\texttt{Event 1}}] \\
&\leq \Pr[\texttt{Event 1}] + \tfrac{1}{2}(1 - \Pr[\texttt{Event 1}]) = \tfrac{1}{2} + \tfrac{1}{2}\Pr[\texttt{Event 1}]. \\
\Pr[\mathcal{A}\text{ wins}] &\geq \Pr[\mathcal{A}\text{ wins}|\overline{\texttt{Event 1}}]\Pr[\overline{\texttt{Event 1}}] \\
&= \tfrac{1}{2}(1 - \Pr[\texttt{Event 1}]) = \tfrac{1}{2} - \tfrac{1}{2}\Pr[\texttt{Event 1}].
\end{aligned}$$

So we have $\Pr[\texttt{Event 1}] \geq \epsilon(k)$. Now we estimate the probability of Event 2. In the game, $\mathcal{A}$ will notice the difference between the simulation and the real world only if $\mathcal{B}$ rejects a valid $Decrypt^S$ query or $\mathcal{A}$ finds that C^* is an invalid challenge ciphertext. As argued above, the latter event happens only if Event 1 occurs. So we only investigate the rejection of valid decryption query which occurs when

- Case 1. $\mathcal{A}$ queries $Decrypt^S(\texttt{ID}_i, \langle C_1^i, C_2^i\rangle)$ such that $C_1^i = \mathcal{E}(M_{\texttt{pk}}, \texttt{ID}_i, P_i, \sigma_i;\ G_1(m_i, \sigma_i, \texttt{ID}_i, P_i))$ and $C_2^i = G_2(\sigma_i) \oplus m_i$ without querying $G_1(m_i, \sigma_i, \texttt{ID}_i, P_i)$ where P_i is the public key currently associated with $\texttt{ID}_i$, or

- Case 2. $\mathcal{A}$ queries $Decrypt^S(\texttt{ID}_i, \langle C_1^i, C_2^i\rangle)$ such that there are at least two tuples $(m_a, \sigma_a, \texttt{ID}_i, P_i, r_a, C_1^i, C_2^i)$ and $(m_b, \sigma_b, \texttt{ID}_i, P_i, r_b, C_1^i, C_2^i)$ in G_1^{list}. First this case cannot happen if $\sigma_a = \sigma_b$; otherwise $m_a = G_2(\sigma_a) \oplus C_2^i = G_2(\sigma_b) \oplus C_2^i = m_b$, and $(m_a, \sigma_a, \texttt{ID}_i, P_i)$ uniquely defines a tuple in G_1^{list}. Hence Case 2 happens only if $\sigma_a \neq \sigma_b$ which implies $C_1^i = \mathcal{E}(M_{\texttt{pk}}, \texttt{ID}_i, P_i, \sigma_a;\ r_a) = \mathcal{E}(M_{\texttt{pk}}, \texttt{ID}_i, P_i, \sigma_b;\ r_b)$.

Case 1 happens with probability at most γ because $\mathcal{E}$ is γ-uniform and G_1 is truly random so $G_1(m_i, \sigma_i, \texttt{ID}_i, P_i)$ is valid for $(M_{\texttt{pk}}, \texttt{ID}_i, P_i, \sigma_i, C_1^i)$ with probability at most γ.

Now we consider the probability of Case 2. Because $\mathcal{E}$ is γ-uniform and G_1 is truly random so one query $G_1(m_b, \sigma_b, \texttt{ID}_i, P_i)$ is valid for $(M_{\texttt{pk}}, \texttt{ID}_i, P_i, \sigma_b, C_1^i)$ with probability at most γ where C_1^i is determined by $C_1^i = \mathcal{E}(M_{\texttt{pk}}, \texttt{ID}_i, P_i, \sigma_a;\ r_a)$. Note that for a fixed C_2^i there are $2^{|\mathbb{M}\,\overline{\Pi}(\overline{M_{\texttt{pk}}})|}$ pairs of (m_b, σ_b). For q_{G_1} queries ($q_{G_1} < 2^{|\mathbb{M}\,\overline{\Pi}(\overline{M_{\texttt{pk}}})|}$), Case 2 happens with probability at most $1 - (1-\gamma)^{q_{G_1}}$.

And if Case 1 happens, Case 2 won't happen. It comes that

$$\Pr[\overline{\texttt{Event 2}}] \geq (1 - \max\{\gamma, 1 - (1-\gamma)^{q_{G_1}}\})^{q_D} \geq (1-\gamma)^{q_{G_1} \cdot q_D}.$$

Overall, we have that

$$\begin{aligned}
\texttt{Adv}_{\mathcal{B}}(k) &\geq \tfrac{1}{q_{G_1}+q_{G_2}}\Pr[\overline{\texttt{Event 2}}]\Pr[\texttt{Event 1}] \\
&\geq \tfrac{1}{q_{G_1}+q_{G_2}}(1-\gamma)^{q_{G_1} q_D} \cdot \epsilon(k).
\end{aligned}$$

□

In Section 4.5.4, we will see that including the message recipient's identifier and public key in G_1 is necessary for the construction to work in general.

4.5.2 The Concrete Scheme

The CL-PKE1 scheme is based on BF-IBE, a full domain hash IBE with pairing. The scheme consists of following algorithms:

CL.Gen(1^k). On input 1^k, the algorithm works as follows.

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ of prime order p, an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$. Pick a random generator $P_2 \in \mathbb{G}_2$ and set $P_1 = \psi(P_2)$. Note that the implementation of the scheme does not need the isomorphism ψ, hence P_1 can be a random generator of $\mathbb{G}_1$.

2. Pick a random $s \in \mathbb{Z}_p$ and compute $P_{pub} = sP_1$.

3. Pick four cryptographic hash functions:

$$H_1 : \{0,1\}^* \rightarrow \mathbb{G}_2,$$
$$H_2 : \mathbb{G}_t \times \mathbb{G}_1 \rightarrow \{0,1\}^n,$$
$$H_3 : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p,$$
$$H_4 : \{0,1\}^n \rightarrow \{0,1\}^n,$$

   for some integer $n > 0$.

4. Output the master public key $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, H_1, H_2, H_3, H_4)$ and the master secret key $M_{\mathtt{sk}} = s$.

The message space is $\mathbb{M} = \{0,1\}^n$, the ciphertext space is $\mathbb{C} = \mathbb{G}_1 \times \{0,1\}^n \times \{0,1\}^n$ and the randomness space is $\mathbb{R} = \{0,1\}^n$.

CL.PartialKey$(M_{\mathtt{sk}}, M_{\mathtt{pk}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathtt{pk}}$ and $M_{\mathtt{sk}}$, the algorithm computes $Q_A = H_1(\mathtt{ID}_A) \in \mathbb{G}_2$, $D_A = sQ_A$ and returns D_A.

CL.SecretVal$(M_{\mathtt{pk}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A$ and $M_{\mathtt{pk}}$, the algorithm outputs a random $X_A \in \mathbb{Z}_p$.

CL.PrivateKey$(M_{\mathtt{pk}}, D_A, X_A)$. Given $M_{\mathtt{pk}}$, D_A and X_A, the algorithm outputs $S_A = (D_A, X_A)$.

CL.PublicKey$(M_{\mathfrak{pk}}, X_A, \mathtt{ID}_A)$. Given $M_{\mathfrak{pk}}$ and X_A, the algorithm outputs $P_A = X_A P_1$.

CL.Encrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, m)$. Given a plaintext $m \in \{0,1\}^n$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathfrak{pk}}$ and the public key P_A of the entity, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, m, \mathtt{ID}_A, P_A)$.
2. Compute $Q_A = H_1(\mathtt{ID}_A)$, $\xi = \hat{e}(P_{pub}, Q_A)^r$ and $f = rP_A$.
3. Set the ciphertext to $C = \langle rP_1, \sigma \oplus H_2(\xi, f), m \oplus H_4(\sigma)\rangle$.

CL.Decrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle U, V, W\rangle \in \mathbb{C}$, the private key $S_A = (D_A, X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm takes the following steps:

1. Compute $\xi' = \hat{e}(U, D_A)$, $f' = X_A U$ and $\sigma' = V \oplus H_2(\xi', f')$.
2. Compute $m' = W \oplus H_4(\sigma')$ and $r' = H_3(\sigma', m', \mathtt{ID}_A, P_A)$.
3. If $U \neq r'P_1$, output $\perp$, else return m' as the plaintext.

4.5.3 Security Analysis of CL-PKE1

To prove the security of CL-PKE1, we will first prove that the following Basic-CL-PKE1 is CL-OW-CPA secure, then apply Theorem 4.5.1 to obtain the security result.

Basic-CL-PKE1 shares most of the algorithms with CL-PKE1 except the following two (the hash functions H_3 and H_4 in **CL.Gen** of CL-PKE1 are unnecessary in Basci-CL-PKE1).

CL.Encrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, m;\ r)$. Given a plaintext $m \in \{0,1\}^n$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathfrak{pk}}$ and the public key P_A of the entity, the following steps are performed.

1. Compute $Q_A = H_1(\mathtt{ID}_A)$, $\xi = \hat{e}(P_{pub}, Q_A)^r$ and $f = rP_A$.

2. Set the ciphertext to $C = \langle rP_1, m \oplus H_2(\xi, f)\rangle$.

CL.Decrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle U, V\rangle$, the private key $S_A = (D_A, X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm takes the following steps:

1. Compute $\xi' = \hat{e}(U, D_A)$ and $f' = X_A U$.

2. Return $m' = V \oplus H_2(\xi', f')$ as the plaintext.

Lemma 4.5.2. *Basic-CL-PKE1 is secure in the sense of CL-OW-CPA against Type-I adversaries provided that H_1 and H_2 are modeled as random oracles and the $BDH^{\psi}_{2,2,2}$ assumption is sound. Specifically, assume there exists a Type-I adversary $\mathcal{A}$ breaks Basic-CL-PKE1 with CL-OW-CPA attacks with advantage $\epsilon(k)$, and in the attack $\mathcal{A}$ runs in time $t(k)$ and makes q_{H_1} and q_{H_2} queries on H_1 and H_2 respectively. Then there exists an algorithm $\mathcal{B}$ to solve the $BDH^{\psi}_{2,2,2}$ problem with advantage and time as follows:*

$$\begin{aligned} Adv_{\mathcal{B}}^{BDH^{\psi}_{2,2,2}} &\geq (1 - \tfrac{q_{H_2}}{2^n})\epsilon(k)/q_{H_1}, \\ t_{\mathcal{B}}(k) &\leq t(k) + O(q_{H_2} \cdot \tau), \end{aligned}$$

where τ is the time of a pairing.

Proof: Given a $\mathrm{BDH}^{\psi}_{2,2,2}$ challenge (aP_2, bP_2, cP_2) with pairing parameters. Algorithm $\mathcal{B}$ simulates algorithm **CL.Gen** of Basic-CL-PKE1 to create the master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, \psi(aP_2), H_1, H_2)$ and sets the master secret key $M_{\mathfrak{sk}} = a$ which $\mathcal{B}$ does not know. H_1, H_2 are random oracles controlled by $\mathcal{B}$. Algorithm $\mathcal{B}$ passes $M_{\mathfrak{pk}}$ to $\mathcal{A}$ and randomly selects $1 \leq I \leq q_{H_1}$ and responds to queries as follows (for simplicity, we assume that the adversary abides by the rules of the CL-OW-CPA game).

- $H_1(\mathtt{ID}_i)$: $\mathcal{B}$ maintains a list H_1^{list} of tuples $\langle \mathtt{ID}_j, Q_j, h_j\rangle$ as explained below. The list is initially empty. When $\mathcal{A}$ makes a query at a point $\mathtt{ID}_i$, $\mathcal{B}$ responds as follows:
  - If $\mathtt{ID}_i$ already appears on H_1^{list} in a tuple $\langle \mathtt{ID}_i, Q_i, h_i\rangle$, then $\mathcal{B}$ responds with Q_i.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $\langle ID_I, bP_2, \perp\rangle$ into the tuple list and responds with $H_1(\mathtt{ID}_I) = bP_2$.
  - Otherwise, $\mathcal{B}$ selects a random integer $h_i \in \mathbb{Z}_p$, computes $Q_i = h_i P_2$, and stores $\langle \mathtt{ID}_i, Q_i, h_i\rangle$ into the tuple list. $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = Q_i$.
- **Public-Key-Broadcast**$(\mathtt{ID}_i)$: $\mathcal{B}$ maintains a list P^{list} with tuples of $\langle \mathtt{ID}_i, U_i, X_i\rangle$ and responds to the queries as follows:
  - If a tuple exists for $\mathtt{ID}_i$, then $\mathcal{B}$ returns the corresponding U_i.
  - Otherwise, $\mathcal{B}$ randomly samples $X_i \in \mathbb{Z}_p$ and inserts a tuple $\langle \mathtt{ID}_i, X_i P_1, X_i\rangle$ into the list and returns $X_i P_1$.

- **Public-Key-Replace**($\mathtt{ID}_i, P$): $\mathcal{B}$ replaces the tuple on P^{list} for $\mathtt{ID}_i$ with a new tuple $\langle \mathtt{ID}_i, P, \perp \rangle$.
- **Partial-Private-Key-Extract**($\mathtt{ID}_i$): $\mathcal{B}$ first searches H_1^{list} for the tuple with $\mathtt{ID}_i$. If no such a tuple is found, then $H_1(\mathtt{ID}_i)$ is queried. If in the found tuple $h_i = \perp$, then $\mathcal{B}$ aborts the game. Otherwise, $\mathcal{B}$ returns $h_i a P_2$.
- **Secret-Value-Extract**($\mathtt{ID}_i$): $\mathcal{B}$ searches P^{list} for $\mathtt{ID}_i$, if no tuple is found, then *Public-Key-Broadcast*($\mathtt{ID}_i$) is queried first. $\mathcal{B}$ returns X_i found on P^{list} with $\mathtt{ID}_i$.
- $H_2(K_i, F_i)$: To respond to the query, $\mathcal{B}$ maintains a list H_2^{list} with tuples of the form $\langle K_i, F_i, \zeta_i \rangle$. On the query, $\mathcal{B}$ does the following operations:
  - If a tuple (K_i, F_i, ζ_i) is on the list, then $\mathcal{B}$ responds with ζ_i.
  - Otherwise, $\mathcal{B}$ randomly chooses $\zeta_i \in \{0,1\}^n$ and adds the tuple $\langle K_i, F_i, \zeta_i \rangle$ to the list. It responds to $\mathcal{A}$ with ζ_i.
- **Challenge:** Once $\mathcal{A}$ decides that Phase 1 of the game is over, it outputs $\mathtt{ID}^*$ on which it wishes to be challenged. $\mathcal{B}$ queries $H_1(\mathtt{ID}^*)$ and if h^* on H_1^{list} for $\mathtt{ID}^*$ is not $\perp$, then $\mathcal{B}$ aborts the game (for simplicity). Otherwise $\mathcal{B}$ randomly samples $C_2^* \in \{0,1\}^n$ and returns $C^* = (\psi(cP_2), C_2^*)$ as the challenge ciphertext. Note that the plaintext m^* of the C^* is
$$m^* = C_2^* \oplus H_2(\hat{e}(\psi(cP_2), abP_2), cP^*)$$
where P^* is the current public key on P^{list} for $\mathtt{ID}^*$.
- **Guess:** Once $\mathcal{A}$ decides Phase 2 of the game is over, it outputs m'. $\mathcal{B}$ does the following to respond to the BDH challenge.
  - $\mathcal{B}$ computes $\zeta^* = m' \oplus C_2^*$.
  - $\mathcal{B}$ goes through H_2^{list} to find tuples with $\langle K_i, F_i, \zeta^* \rangle$ with $\hat{e}(F_i, P_2) = \hat{e}(P^*, cP_2)$.
  - If no such a tuple or more than one tuple is found, $\mathcal{B}$ fails the game.
  - $\mathcal{B}$ returns K_i from the found tuple to the BDH challenge.

We define following events: **Event 1** is that $\mathcal{B}$ aborts the game prematurely. **Event 2** is that $H_2(\hat{e}(P_1, P_2)^{abc}, cP^*)$ is queried at some point during the simulation above. **Event 3** is that more than one tuple $\langle K_i, F_i, \zeta^* \rangle$ with $\hat{e}(F_i, P_2) = \hat{e}(P^*, cP_2)$ is found in H_2^{list}.

Through a standard argument, we have

$$\begin{aligned} \Pr[\overline{\texttt{Event 1}}] &\geq 1/q_{H_1} \\ \Pr[\texttt{Event 2}] &\geq \epsilon(k) \\ \Pr[\texttt{Event 3}] &\leq q_{H_2}/2^n \end{aligned}$$

Overall we have

$$\mathtt{Adv}_{\mathcal{B}}^{\mathtt{BDH}_{2,2,2}}(k) \geq \Pr[\overline{\texttt{Event 1}} \wedge \texttt{Event 2} \wedge \overline{\texttt{Event 3}}] \approx (1 - q_{H_2}/2^n)\epsilon(k)/q_{H_1}.$$

□

Lemma 4.5.3. *Basic-CL-PKE1 is secure in the sense of CL-OW-CPA against Type-II adversaries provided that H_2 is modeled as random oracle and the $DH^{\psi}_{1,2,1}$ assumption is sound. Specifically, assume there exists a Type-II adversary $\mathcal{A}$ breaks Basic-CL-PKE1 with CL-OW-CPA attacks with advantage $\epsilon(k)$, and in the attack $\mathcal{A}$ runs in time $t(k)$ and gets q_P entity public keys. Then there exists an algorithm $\mathcal{B}$ to solve the $DH^{\psi}_{1,2,1}$ problem with advantage and time as follows:*

$$\begin{aligned} Adv_{\mathcal{B}}^{DH^{\psi}_{1,2,1}} &\geq \epsilon(k)/q_P, \\ t_{\mathcal{B}}(k) &\leq t(k) + O(q_{H_2} \cdot \tau), \end{aligned}$$

where τ is the time of a pairing.

Proof: Given a $\mathrm{DH}^{\psi}_{1,2,1}$ challenge (aP_1, bP_2) with pairing parameters, algorithm $\mathcal{B}$ simulates algorithm **CL.Gen** of Basic-CL-PKE1 to create the master public key $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, sP_1, H_1, H_2)$ and sets the master secret key $M_{\mathtt{sk}} = s$ which is randomly sampled by $\mathcal{B}$ from $\mathbb{Z}_p$. H_1 is a hash function and H_2 is a random oracle controlled by $\mathcal{B}$. Algorithm $\mathcal{B}$ passes $M_{\mathtt{pk}}$ with $M_{\mathtt{sk}}$ to $\mathcal{A}$ and randomly selects $1 \leq I \leq q_P$ and responds to queries as follows (for simplicity, we assume that the adversary abides by the rules of the CL-OW-CPA game).

- $H_2(K_i, F_i)$**:** To respond to the query, $\mathcal{B}$ maintains a list H_2^{list} with tuples of the form $\langle K_i, F_i, \zeta_i \rangle$. On the query, $\mathcal{B}$ does the following operations:
  - If a tuple (K_i, F_i, ζ_i) is on the list, then $\mathcal{B}$ responds with ζ_i.
  - Otherwise, $\mathcal{B}$ randomly chooses $\zeta_i \in \{0,1\}^n$ and adds the tuple $\langle K_i, F_i, \zeta_i \rangle$ to the list. It responds to $\mathcal{A}$ with ζ_i.
- **Public-Key-Broadcast(**$\mathtt{ID}_i$**):** $\mathcal{B}$ maintains a list P^{list} with tuples of $\langle \mathtt{ID}_i, U_i, X_i \rangle$ and responds to the queries as follows:
  - If a tuple exists for $\mathtt{ID}_i$, then $\mathcal{B}$ returns the corresponding U_i.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $\langle ID_I, aP_1, \perp \rangle$ into the tuple list and responds with aP_1.
  - Otherwise, $\mathcal{B}$ randomly samples $X_i \in \mathbb{Z}_p$ and inserts a tuple $\langle \mathtt{ID}_i, X_iP_1, X_i \rangle$ into the list and returns X_iP_1.
- **Public-Key-Replace(**$\mathtt{ID}_i, P$**):** $\mathcal{B}$ replaces the tuple on P^{list} for $\mathtt{ID}_i$ with a new tuple $\langle \mathtt{ID}_i, P, \perp \rangle$.
- **Secret-Value-Extract(**$\mathtt{ID}_i$**):** $\mathcal{B}$ searches P^{list} for $\mathtt{ID}_i$, and if no tuple is found, then *Public-Key-Broadcast*($\mathtt{ID}_i$) is queried first. $\mathcal{B}$ returns X_i found on P^{list} with $\mathtt{ID}_i$.
- **Challenge:**. Once $\mathcal{A}$ decides that Phase 1 of the game is over, it outputs $\mathtt{ID}^*$ on which it wishes to be challenged. $\mathcal{B}$ queries *Public-Key-Broadcast*($\mathtt{ID}_i$) and if $U^* \neq aP_1$ on P^{list} for $\mathtt{ID}^*$, then $\mathcal{B}$ aborts the game (for simplicity). $\mathcal{B}$ randomly sample $C_2^* \in \{0,1\}^n$

and returns $C^* = (\psi(bP_2), C_2^*)$ as the challenge ciphertext. Note that the plaintext m^* of the C^* is

$$m^* = C_2^* \oplus H_2(\hat{e}(\psi(bP_2), sH_1(\mathtt{ID}^*)), abP_1).$$

- **Guess:** Once $\mathcal{A}$ decides Phase 2 of the game is over, it outputs m'. $\mathcal{B}$ computes $\zeta^* = m' \oplus C_2^*$ and goes through H_2^{list} to find a tuple of $\langle \hat{e}(bP_1, sH_1(\mathtt{ID}^*), F_i, \zeta^* \rangle$ with $\hat{e}(F_i, P_2) = \hat{e}(aP_1, bP_2)$. If no such a tuple is found, then $\mathcal{B}$ fails the game. Otherwise $\mathcal{B}$ responds to the DH challenge with F_i.

Similar to Lemma 4.5.2 we define following events: **Event 1** is that $\mathcal{B}$ aborts the game prematurely. **Event 2** is that $H_2(\hat{e}(bP_1, sH_1(\mathtt{ID}^*), abP_1)$ is queried at some point during the simulation above.

Through a standard argument, we have

$$\begin{aligned} &\Pr[\overline{\texttt{Event 1}}] \geq 1/q_P \\ &\Pr[\texttt{Event 2}] \geq \epsilon(k) \end{aligned}$$

Overall we have

$$\mathtt{Adv}_{\mathcal{B}}^{\mathtt{DH}_{1,2}}(k) \geq \Pr[\overline{\texttt{Event 1}} \wedge \texttt{Event 2}] = \epsilon(k)/q_P.$$

□

Following from Theorem 4.5.1 and Lemma 4.5.2 and 4.5.3, we have the following security result of CL-PKE1. Note that Basic-CL-PKE1 is $\frac{1}{2^n}$-uniform.

Theorem 4.5.4. *CL-PKE1 is secure against Type-I adversary with CL-IND-CCA2 attacks provided H_i $(1 \leq i \leq 4)$ are modeled as random oracles and the $BDH^{\psi}_{2,2,2}$ assumption is sound. CL-PKE1 is secure against Type-II adversary with CL-IND-CCA2 attacks provided H_i $(2 \leq i \leq 4)$ are modeled as random oracles and the $DH^{\psi}_{1,2,1}$ assumption is sound.*

Specifically, assume a Type-I adversary $\mathcal{A}_I$ breaks CL-PKE1 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$ and in the attack $\mathcal{A}_I$ makes q_D decryption queries and q_i queries on H_i for $1 \leq i \leq 4$ and $q_3 < 2^n$, then there exists an algorithm $\mathcal{B}_I$ to solve the $BDH^{\psi}_{2,2,2}$ problem with following advantage and time

$$\begin{aligned} Adv_{\mathcal{B}_I}^{BDH^{\psi}_{2,2,2}}(k) &\geq \tfrac{(1-q_2/2^n)\epsilon(k)}{q_1(q_3+q_4)}(1-\tfrac{1}{2^n})^{q_3 q_D}, \\ t_{\mathcal{B}_I}(k) &\leq t(k) + O(q_3 t_{\mathcal{E}} + q_2 \tau). \end{aligned}$$

where $t_{\mathcal{E}}$ is the cost of Basic-CL-PKE1 and τ is the time of a pairing.

Assume a Type-II adversary $\mathcal{A}_{II}$ breaks CL-PKE1 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$ and in the attack $\mathcal{A}_{II}$ makes q_P public key queries and q_i queries on H_i for $2 \leq i \leq 4$ and $q_3 < 2^n$, then there exists an algorithm $\mathcal{B}_{II}$ to solve the $DH^{\psi}_{1,2,1}$ problem with following advantage and time

$$\begin{aligned} Adv_{\mathcal{B}_{II}}^{DH^{\psi}_{1,2,1}}(k) &\geq \tfrac{\epsilon(k)}{q_P(q_3+q_4)}(1-\tfrac{1}{2^n})^{q_3 q_D}, \\ t_{\mathcal{B}_{II}}(k) &\leq t(k) + O(q_3 t_{\mathcal{E}} + q_2 \tau). \end{aligned}$$

4.5.4 On the Al-Riyami-Paterson's Second CL-PKE

Independent of our work [46], Al-Riyami and Paterson proposed a CL-PKE [6], which we call AP-CL-PKE2. The scheme shares most the algorithms with CL-PKE1 except CL.Encrypt and CL.Decrypt. And AP-CL-PKE2 uses the hash functions $H_1, H_2', H_2'', H_3', H_4$ where H_1, H_4 are just as of CL-PKE1 and H_2', H_2'', H_3' are defined as follows:

$$H_2' : \mathbb{G}_t \times \rightarrow \{0,1\}^n,$$
$$H_2'' : \mathbb{G}_1 \times \rightarrow \{0,1\}^n,$$
$$H_3' : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_p,$$

AP-CL-PKE2's encryption and decryption algorithm are as follows:

CL.Encrypt$(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, m)$. Given a plaintext $m \in \{0,1\}^n$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathtt{pk}}$ and the public key P_A of the entity, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3'(\sigma, m)$.

2. Compute $Q_A = H_1(\mathtt{ID}_A)$, $\xi = \hat{e}(P_{pub}, Q_A)^r$ and $f = rP_A$.

3. Set the ciphertext to $C = \langle rP_1, \sigma \oplus H_2'(\xi) \oplus H_2''(f), m \oplus H_4(\sigma) \rangle$.

CL.Decrypt$(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle U, V, W \rangle \in \mathbb{C}$, the private key $S_A = (D_A, X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathtt{pk}}$, the algorithm takes the following steps:

1. Compute $\xi' = \hat{e}(U, D_A)$, $f' = X_A U$ and $\sigma' = V \oplus H_2'(\xi') \oplus H_2''(f')$.

2. Compute $m' = W \oplus H_4(\sigma')$ and $r' = H_3'(\sigma', m')$.

3. If $U \neq r'P_1$, output $\perp$, else return m' as the plaintext.

Though AP-CL-PKE2 bears the very similarity with CL-PKE1, the scheme suffers from two attacks: a Type-I^C attack and a Type-II attack.

An adversary $\mathcal{A}$ launches the Type-I^C attack as follows:

1. $\mathcal{A}$ randomly chooses $\mathtt{ID}^*$ and two messages m_0, m_1, and passes them to the challenger $\mathcal{T}$ to get the ciphertext $C^* = \langle U^*, V^*, W^* \rangle$ of m_b for $b \in \{0, 1\}$.

2. $\mathcal{A}$ issues *Secret-Value-Extract*($\mathtt{ID}^*$) to get the secret value X^*.

3. $\mathcal{A}$ randomly chooses $X' \in \mathbb{Z}_p$ and issues *Public-Key-Replace*($\mathtt{ID}^*, X'P_1$).

4. $\mathcal{A}$ generates a ciphertext $C' = \langle U^*, V^* \oplus H_2''(X^*U^*) \oplus H_2''(X'U^*), W^* \rangle$. Note that C' is a valid ciphertext of m_b for $\mathtt{ID}^*$ with the public key $X'P_1$.

5. $\mathcal{A}$ issues $Decrypt^C(\mathtt{ID}^*, C', X')$ to get the plaintext m_b to win Game 1.

Note that in Al-Riyami-Paterson's CL-PKE formulation [6], the oracle *Secret-Value-Extract* does not exist. But the attack still works by $\mathcal{A}$ first choosing $X^* \in Z_p$ and replacing $\mathtt{ID}^*$'s public key with X^*P_1 before issuing the challenge [168]. It is easy to see merely adding the message recipient's identifier in H_3' does not defend the attack.

An adversary $\mathcal{A}$ can launch a trivial Type-II attack as follows:

1. $\mathcal{A}$ issues *Public-Key-Broadcast*($\mathtt{ID}^*$) to get the public key value P^*.

2. $\mathcal{A}$ randomly chooses two messages m_0, m_1, and passes $\mathtt{ID}^*$ and the messages to the challenger $\mathcal{T}$ to get the ciphertext $C^* = \langle U^*, V^*, W^* \rangle$ of m_b for $b \in \{0, 1\}$.

3. $\mathcal{A}$ randomly chooses $\mathtt{ID}'$ and issues *Public-Key-Replace*($\mathtt{ID}', P^*$).

4. $\mathcal{A}$ generates a ciphertext $C' = \langle U^*, V^* \oplus H_2'(\xi') \oplus H_2'(\xi''), W^* \rangle$ with $\xi' = \hat{e}(U^*, D_{\mathtt{ID}^*})$ and $\xi'' = \hat{e}(U^*, D_{\mathtt{ID}'})$ where $D_{\mathtt{ID}^*}, D_{\mathtt{ID}'}$ are the partial keys of $\mathtt{ID}^*, \mathtt{ID}'$ respectively. $\mathcal{A}$ can compute both values with the master secret key $M_{\mathtt{st}}$. Note that C' is a valid ciphertext of m_b for $\mathtt{ID}'$ with the public key P^*.

5. $\mathcal{A}$ issues $Decrypt^S(\mathtt{ID}', C', X')$ to get the plaintext m_b to win Game 2.

Note that the Al-Riyami-Paterson's CL-PKE formulation [5, 6] does not consider this type of attack. It is easy to find out that merely adding the message recipient's public key in H_3' does not prevent the attack.

By applying Theorem 4.5.1 it is straightforward to prove that including the message recipient's identifier and public key in H_3' as CL-PKE1 does, the modified scheme is secure.

4.6 CL-PKE2

In this section, we construct a CL-PKE (referred to as CL-PKE2) strictly following the heuristic approach in Section 4.4. The scheme is based on BB_1-IBE [9], a commutative blinding IBE.

4.6.1 The Scheme

CL-PKE2 consists of following algorithms:

CL.Gen(1^k): On input 1^k, the algorithm works as follows:

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ of prime order p and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$. Pick random generator $P_2 \in \mathbb{G}_2$ and $P_1 \in \mathbb{G}_1$.

2. Randomly sample a, b and $c \in \mathbb{Z}_p$. Set $Q_1 = aP_1, Q_2 = bP_1, Q_3 = cP_1 \in \mathbb{G}_1$, and $\hat{Q}_1 = aP_2, \hat{Q}_2 = bP_2, \hat{Q}_3 = cP_2 \in \mathbb{G}_2$. Compute $v_0 = \hat{e}(Q_1, \hat{Q}_2) = \hat{e}(P_1, P_2)^{ab}$.

3. Pick three cryptographic hash functions:

$$H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p,$$
$$H_2 : \mathbb{G}_t \times \mathbb{G}_1 \rightarrow \{0,1\}^n,$$
$$H_3 : \mathbb{G}_t \times \mathbb{G}_1 \times \{0,1\}^n \times \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p,$$

   for some integer $n > 0$.

4. Output the master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, n, P_1, Q_1, Q_3, v_0, H_1, H_2, H_3)$ and the master secret key $M_{\mathfrak{sk}} = (P_2, a, b, c)$.

The message space is $\mathbb{M} = \{0,1\}^n$, the ciphertext space is $\mathbb{C} = \{0,1\}^n \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{Z}_p$ and the randomness space is $\mathbb{R} = \mathbb{Z}_p$.

CL.PartialKey$(M_{\mathfrak{sk}}, M_{\mathfrak{pk}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathfrak{pk}}$ and $M_{\mathfrak{sk}}$, the algorithm randomly picks $t \in \mathbb{Z}_p$ and outputs

$$D_A = (D_1, D_2) = ((ab + (aH_1(\mathtt{ID}_A) + c)t)P_2, tP_2).$$

We note that P_2 in $M_{\mathfrak{sk}}$ can be disclosed, in particular, on Type-1 pairings $P_1 = P_2$. Given $M_{\mathfrak{pk}}$ with P_2, one can verify if D_A is a signature on $\mathtt{ID}_A$.

CL.SecretVal$(M_{\mathfrak{pk}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm returns a random $X_A \in \mathbb{Z}_p$.

CL.PrivateKey$(M_{\mathfrak{pk}}, D_A, X_A)$. Given $M_{\mathfrak{pk}}$, D_A and X_A, the algorithm outputs $S_A = (D_A, X_A)$.

CL.PublicKey$(M_{\mathfrak{pk}}, X_A, \mathtt{ID}_A)$. Given $M_{\mathfrak{pk}}$ and X_A, the algorithm outputs $P_A = X_A P_1$.

CL.Encrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, m)$. Given a plaintext $m \in \mathbb{M}$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathfrak{pk}}$ and the public key P_A of the entity, the following steps are performed.

1. Pick a random $r \in \mathbb{Z}_p$ and compute $C_1 = rP_1$ and $C_2 = rQ_3 + rH_1(\mathtt{ID}_A)Q_1$.
2. Compute $\xi = v_0^r, f = rP_A$ and $C_0 = m \oplus H_2(\xi, f)$.
3. Compute $\sigma = r + H_3(\xi, f, C_0, C_1, C_2) \mod p$.
4. Set the ciphertext to $C = \langle C_0, C_1, C_2, \sigma \rangle$.

CL.Decrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle C_0, C_1, C_2, \sigma \rangle \in \mathbb{C}$, the private key $S_A = (D_A = (D_1, D_2), X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm takes the following steps:

1. Compute $\xi' = \frac{\hat{e}(C_1, D_1)}{\hat{e}(C_2, D_2)}$ and $f' = X_A C_1$.

2. Compute $r' = \sigma - H_3(\xi', f', C_0, C_1, C_2) \mod p$, output $\perp$ if $(\xi', C_1) \neq (v_0^{r'}, r'P_1)$.

3. Compute $m' = C_0 \oplus H_2(\xi', f')$ and return m' as the plaintext.

4.6.2 Security Analysis of CL-PKE2

Before we formally analyse the security of CL-PKE2, we generalise the BDH assumption as follows:

Assumption 4.6.1. ***(General BDH)*** *For $a, b, c \in_R \mathbb{Z}_p$, given (aP_1, cP_1, aP_2, bP_2) with the pairing parameters, computing $\hat{e}(P_1, P_2)^{abc}$ is hard.*

We note that if an efficient isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ exists, then the $\text{BDH}_{i,2,2}$ assumption for $i \in \{1, 2\}$ implies the general BDH assumption.

CL-PKE2's security is summarised by the following theorem.

Theorem 4.6.1. *CL-PKE2 is secure against Type-I adversary with CL-IND-CCA2 attacks provided H_i $(1 \leq i \leq 3)$ are modeled as random oracles and the general BDH assumption is sound. CL-PKE2 is secure against Type-II adversary with CL-IND-CCA2 attacks provided H_i $(i = 2, 3)$ are modeled as random oracles and the $DH_{1,1,1}$ assumption is sound.*

Specifically, assume a Type-I adversary $\mathcal{A}_I$ breaks CL-PKE2 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$ and in the attack $\mathcal{A}_I$ makes q_D decryption queries and q_i queries on H_i for $1 \leq i \leq 3$, then there exists an algorithm $\mathcal{B}_I$ to solve the general BDH problem with following advantage and time

$$\begin{aligned} Adv_{\mathcal{B}_I}^{General\ BDH}(k) &\geq \frac{(1-q_1/p)^{q_1}\epsilon(k)}{q_1(q_2+q_3)}(1-\frac{1}{p})^{q_D}, \\ t_{\mathcal{B}_I}(k) &\leq t(k) + O(q_1 t_2 + q_D(t_1 + t_3)), \end{aligned}$$

where t_i for $i = 1, 2, 3$ is the cost of operation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ respectively.

Assume a Type-II adversary $\mathcal{A}_{II}$ breaks CL-PKE2 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$, and in the attack $\mathcal{A}_{II}$ makes q_P public key queries and q_i queries on H_i for $i = 2, 3$, then there exists an algorithm $\mathcal{B}_{II}$ to solve the $DH_{1,1,1}$ problem with following advantage and time

$$\begin{aligned} Adv_{\mathcal{B}_{II}}^{DH_{1,1,1}}(k) &\geq \frac{\epsilon(k)}{q_P(q_2+q_3)}(1-\frac{1}{p})^{q_D}, \\ t_{\mathcal{B}_{II}}(k) &\leq t(k) + O(q_D t_1). \end{aligned}$$

Proof: We first deal with the Type-I adversary $\mathcal{A}_I$. Given a general BDH problem $(P_1, aP_1, rP_1,\ P_2, aP_2, bP_2)$, we can construct an algorithm $\mathcal{B}_I$ to compute $\hat{e}(P_1, P_2)^{abr}$ by making use of $\mathcal{A}_I$ as a subroutine. $\mathcal{B}_I$ randomly chooses $1 \leq I \leq q_1$ and starts to simulate the CL-IND-CCA2 Game 1 as follows:

- **CL.Gen:** $\mathcal{B}_I$ simulates **CL.Gen** as follows:
  - Set $Q_1 = aP_1, \hat{Q}_1 = aP_2, \hat{Q}_2 = bP_2$,
  - Randomly sample $u, h \in \mathbb{Z}_p$ and set $c = h - ua \mod p$ and compute $Q_3 = hP_1 - uaP_1 = cP_1$ and $\hat{Q}_3 = hP_2 - uaP_2 = cP_2$.
  - Compute $v_0 = \hat{e}(aP_1, bP_2) = \hat{e}(P_1, P_2)^{ab}$.
  - Output the master public key $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, n, P_1, Q_1, Q_3, v_0, H_1, H_2, H_3)$ where H_i are random oracles controlled by $\mathcal{B}_I$ and set the master secret key $M_{\mathtt{sk}} = (P_2, a, b, c)$.
- $H_1(\mathtt{ID}_i)$: $\mathcal{B}_I$ maintains a list H_1^{list} of tuples $\langle \mathtt{ID}_j, u_j \rangle$ as explained below. The list is initially empty. When $\mathcal{A}_I$ makes a query on $\mathtt{ID}_i$, $\mathcal{B}_I$ responds as follows:
  - If $\mathtt{ID}_i$ already appears on H_1^{list} in a tuple $\langle \mathtt{ID}_i, u_i \rangle$, then $\mathcal{B}_I$ responds with u_i.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}_I$ stores $\langle \mathtt{ID}_I, u \rangle$ into the tuple list and responds with u.
  - Otherwise, $\mathcal{B}_I$ selects a random integer $u_i \in \mathbb{Z}_p$ and stores $\langle \mathtt{ID}_i, u_i \rangle$ into the tuple list. $\mathcal{B}_I$ responds with u_i.
- **Public-Key-Broadcast**($\mathtt{ID}_i$): $\mathcal{B}_I$ maintains a list P^{list} with tuples of the form $\langle \mathtt{ID}_i, U_i, X_i \rangle$ and responds to the queries as follows:
  - If a tuple exists for $\mathtt{ID}_i$, then $\mathcal{B}_I$ returns the corresponding U_i.
  - Otherwise, $\mathcal{B}_I$ randomly samples $X_i \in \mathbb{Z}_p$ and inserts a tuple $\langle \mathtt{ID}_i, X_i P_1, X_i \rangle$ into P^{list} and returns $X_i P_1$.
- **Public-Key-Replace**($\mathtt{ID}_i, P$): $\mathcal{B}_I$ replaces the tuple on P^{list} for $\mathtt{ID}_i$ with a new tuple $\langle \mathtt{ID}_i, P, \perp \rangle$.
- **Partial-Private-Key-Extract**($\mathtt{ID}_i$): $\mathcal{B}_I$ responds to the query as follows
  - $\mathcal{B}_I$ first searches H_1^{list} for the tuple with $\mathtt{ID}_i$. If no such a tuple is found, then $H_1(\mathtt{ID}_i)$ is queried. Assume u_i is on the found tuple in H_1^{list} for $\mathtt{ID}_i$.
  - If $u_i = u$, $\mathcal{B}_I$ aborts the game.
  - $\mathcal{B}_I$ randomly samples $t_i \in \mathbb{Z}_p$ and sets $\tilde{t}_i = t_i - b/(u_i - u)$ and computes

$$\begin{aligned} D_1 &= \tfrac{-h}{u_i - u}\hat{Q}_2 + t_i((u_i - u)\hat{Q}_1 + hP_2) \\ &= \tfrac{-h}{u_i - u}bP_2 + \tfrac{b}{u_i - u}((u_i - u)\hat{Q}_1 + hP_2) + \tilde{t}_i((u_i - u)\hat{Q}_1 + hP_2) \\ &= abP_2 + \tilde{t}_i(u_i aP_2 + cP_2) = (ab + (aH_1(\mathtt{ID}_i) + c)\tilde{t}_i)P_2 \\ D_2 &= t_i P_2 - \tfrac{1}{u_i - u}bP_2 = \tilde{t}_i P_2 \end{aligned}$$

    $\mathcal{B}_I$ outputs (D_1, D_2) as the answer.
- **Secret-Value-Extract**($\mathtt{ID}_i$): $\mathcal{B}_I$ searches P^{list} for $\mathtt{ID}_i$, and if no tuple is found, then *Public-Key-Broadcast*($\mathtt{ID}_i$) is queried first. $\mathcal{B}_I$ returns X_i found on P^{list} with $\mathtt{ID}_i$.

- $H_2(K_i, F_i)$: To respond to the query, $\mathcal{B}_I$ maintains a list H_2^{list} with tuples of the form $\langle K_i, F_i, \zeta_i \rangle$. On the query, $\mathcal{B}_I$ does the following operations:
  - If a tuple (K_i, F_i, ζ_i) is on the list, then $\mathcal{B}_I$ responds with ζ_i.
  - Otherwise, $\mathcal{B}_I$ randomly chooses $\zeta_i \in \{0,1\}^n$ and adds the tuple $\langle K_i, F_i, \zeta_i \rangle$ to the list. It responds to $\mathcal{A}_I$ with ζ_i.
- $H_3(K_i, F_i, C_0^i, C_1^i, C_2^i)$: To respond to the query, $\mathcal{B}_I$ maintains a list H_3^{list} with tuples of the form $\langle K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i \rangle$. On the query, $\mathcal{B}_I$ does the following operations:
  - If a tuple $(K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i)$ is on the list, then $\mathcal{B}_I$ responds with ξ_i.
  - Otherwise, $\mathcal{B}_I$ randomly chooses $\xi_i \in \mathbb{Z}_p$ and adds the tuple $\langle K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i \rangle$ to the list. It responds to $\mathcal{A}_I$ with ξ_i.
- **Decrypt**$^S(\mathtt{ID}_i, C_i)$: $\mathcal{B}_I$ takes the following steps to respond to the query
  - $\mathcal{B}_I$ queries the current public key P_i associated with $\mathtt{ID}_i$ by *Public-Key-Broadcast*($\mathtt{ID}_i$).
  - $\mathcal{B}_I$ parses C_i as $\langle C_0^i, C_1^i, C_2^i, \sigma_i \rangle$ and searches H_3^{list} to find tuples $(K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i)$ and puts them into set S_0.
  - If S_0 is empty, then $\mathcal{B}_I$ outputs $\perp$.
  - For each tuple in S_0,
    - Compute $r_i = \sigma_i - \xi_i \mod p$.
    - If $C_1^i = r_i P_1, C_2^i = r_i Q_3 + r_i H_1(\mathtt{ID}_i) Q_1, K_i = v_0^{r_i}$ and $F_i = r_i P_i$, then $\mathcal{B}_I$ returns $C_0 \oplus H_2(K_i, F_i)$.
  - If no tuple is found to pass the above check, then $\mathcal{B}_I$ outputs $\perp$.
- **Challenge:** Once $\mathcal{A}_I$ decides that Phase 1 is over, it outputs identity $\mathtt{ID}^*$ and two messages m_0, m_1 on which it wishes to be challenged.
  - If $H_1(\mathtt{ID}^*) \neq u$, then $\mathcal{B}_I$ aborts the game.
  - $\mathcal{B}_I$ randomly samples $\sigma^* \in Z_p$ and $C_0^* \in \{0,1\}^n$. $\mathcal{B}_I$ sets $C_1^* = rP_1$ and $C_2^* = rcP_1 + ruaP_1 = hrP_1$.
  - $\mathcal{B}_I$ returns $C^* = \langle C_0^*, C_1^*, C_2^*, \sigma^* \rangle$ as the challenge ciphertext.
- **Guess:** Once $\mathcal{A}_I$ outputs its guess b'. $\mathcal{B}_I$ randomly chooses a K_i from H_2^{list} or H_3^{list} and outputs K_i as the answer of the general BDH problem.

Now we analyse $\mathcal{B}_I$'s probability of outputting the correct answer to the general BDH problem. We define three events. **Event 1** is that $\mathcal{B}_I$ aborts the game prematurely which could happen when $\mathtt{ID}^* \neq \mathtt{ID}_I$ or $H_1(\mathtt{ID}_i) = u$ with $\mathtt{ID}_i \neq \mathtt{ID}_I$. **Event 2** is that $H_2(\hat{e}(P_1, P_2)^{abr}, *)$ or $H_3(\hat{e}(P_1, P_2)^{abr}, *, *, *, *)$ is queried at some point during the simulation above. **Event 3** is that in the game $\mathcal{A}_I$ differentiates $\mathcal{B}_I$ from a real world before Event 2 happens.

Through a standard argument, we have

$$\begin{aligned}\Pr[\overline{\texttt{Event 1}}] &\geq (1-q_1/p)^{q_1}/q_1\\ \Pr[\texttt{Event 2}] &\geq \epsilon(k)\\ \Pr[\overline{\texttt{Event 3}}] &\geq (1-1/p)^{q_D}\end{aligned}$$

Overall we have

$$\mathtt{Adv}^{\mathtt{BDH}}_{\mathcal{B}_I}(k) \geq \frac{1}{q_2+q_3}\Pr[\overline{\texttt{Event 1}} \wedge \mathcal{H}_2 \wedge \overline{\texttt{Event 3}}] = \frac{(1-q_1/p)^{q_1}\epsilon(k)}{q_1(q_2+q_3)}(1-1/p)^{q_D}.$$

Now we deal with the Type-II adversary $\mathcal{A}_{II}$. Given a DH problem (P_1, xP_1, yP_1), we can construct an algorithm $\mathcal{B}_{II}$ to compute xyP_1 by making use of $\mathcal{A}_{II}$ as a subroutine. $\mathcal{B}_{II}$ randomly chooses $1 \leq I \leq q_P$ and starts to simulate the CL-IND-CCA2 Game 2 as follows:

- **CL.Gen:** $\mathcal{B}_{II}$ simulates **CL.Gen** as follows:
  - Randomly sample $a, b, c \in \mathbb{Z}_p$ and a generator $P_2 \in \mathbb{G}_2^*$
  - Set $Q_1 = aP_1, \hat{Q}_1 = aP_2, Q_2 = bP_1, \hat{Q}_2 = bP_2, Q_3 = cP_1, \hat{Q}_3 = cP_2$,
  - Compute $v_0 = \hat{e}(aP_1, bP_2) = \hat{e}(P_1, P_2)^{ab}$.
  - Output the master public key $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, n, P_1, Q_1, Q_3, v_0, H_1, H_2, H_3)$ where H_1 is a hash function and H_2 and H_3 are two random oracles controlled by $\mathcal{B}_{II}$ and set the master secret key $M_{\mathtt{sk}} = (P_2, a, b, c)$.
  - Pass both $M_{\mathtt{pk}}$ and $M_{\mathtt{sk}}$ to $\mathcal{A}_{II}$.
- **Public-Key-Broadcast($\mathtt{ID}_i$):** $\mathcal{B}_{II}$ maintains a list P^{list} with tuples of the form $\langle \mathtt{ID}_i, U_i, X_i \rangle$ and responds to the queries as follows:
  - If a tuple exists for $\mathtt{ID}_i$, then $\mathcal{B}_{II}$ returns the corresponding U_i.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}_{II}$ stores $\langle ID_I, xP_1, \perp \rangle$ into the tuple list and responds with xP_1.
  - Otherwise, $\mathcal{B}_{II}$ randomly samples $X_i \in \mathbb{Z}_p$ and inserts a tuple $\langle \mathtt{ID}_i, X_iP_1, X_i \rangle$ into P^{list} and returns X_iP_1.
- **Secret-Value-Extract($\mathtt{ID}_i$):** $\mathcal{B}_{II}$ searches P^{list} for $\mathtt{ID}_i$, and if no tuple is found, then *Public-Key-Broadcast*($\mathtt{ID}_i$) is queried first. $\mathcal{B}_{II}$ returns X_i found on P^{list} with $\mathtt{ID}_i$.
- $H_2(K_i, F_i)$**:** To respond to the query, $\mathcal{B}_{II}$ maintains a list H_2^{list} with tuples of the form $\langle K_i, F_i, \zeta_i \rangle$. On the query, $\mathcal{B}_{II}$ does the following operations:
  - If a tuple (K_i, F_i, ζ_i) is on the list, then $\mathcal{B}_{II}$ responds with ζ_i.
  - Otherwise, $\mathcal{B}_{II}$ randomly chooses $\zeta_i \in \{0,1\}^n$ and adds the tuple $\langle K_i, F_i, \zeta_i \rangle$ to the list. It responds to $\mathcal{A}_{II}$ with ζ_i.

- $H_3(K_i, F_i, C_0^i, C_1^i, C_2^i)$: To respond to the query, $\mathcal{B}_{II}$ maintains a list H_3^{list} with tuples of the form $\langle K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i \rangle$. On the query, $\mathcal{B}_{II}$ does the following operations:
  - If a tuple $(K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i)$ is on the list, then $\mathcal{B}_{II}$ responds with ξ_i.
  - Otherwise, $\mathcal{B}_{II}$ randomly chooses $\xi_i \in \mathbb{Z}_p$ and adds the tuple $\langle K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i \rangle$ to the list. It responds to $\mathcal{A}_{II}$ with ξ_i.
- **Decrypt**$^S(\mathtt{ID}_i, C_i)$: $\mathcal{B}_{II}$ takes the following steps to respond to the query
  - $\mathcal{B}_{II}$ queries the public key P_i associated with $\mathtt{ID}_i$ by *Public-Key-Broadcast*($\mathtt{ID}_i$).
  - $\mathcal{B}_{II}$ parses C_i as $\langle C_0^i, C_1^i, C_2^i, \sigma_i \rangle$ and searches H_3^{list} to find tuples $(K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i)$ where $K_i = \frac{\hat{e}(C_1^i, D_1^i)}{\hat{e}(C_2^i, D_2^i)}$ with the partial key (D_1^i, D_2^i) computed by algorithm **CL.PartialKey** with $M_{\mathtt{st}}$ and $\mathtt{ID}_i$, and puts them into set S_0.
  - If S_0 is empty, then $\mathcal{B}_{II}$ outputs $\perp$.
  - For each tuple in S_0,
    - Compute $r_i = \sigma_i - \xi_i \mod p$.
    - If $C_1^i = r_i P_1, K_i = v_0^{r_i}$ and $F_i = r_i P_i$, then $\mathcal{B}_{II}$ returns $C_0 \oplus H_2(K_i, F_i)$. Note that given C_1^i, D_1^i, D_2^i and K_i, C_2^i is uniquely determined.
  - If no tuple is found to pass the above check, then $\mathcal{B}_{II}$ outputs $\perp$.
- **Challenge:** Once $\mathcal{A}_{II}$ decides that Phase 1 is over, it outputs identity $\mathtt{ID}^*$ and two messages m_0, m_1 on which it wishes to be challenged.
  - $\mathcal{B}_{II}$ queries *Public-Key-Broadcast*($\mathtt{ID}^*$) and if $U^* \neq xP_1$ on P^{list} for $\mathtt{ID}^*$, then $\mathcal{B}_{II}$ aborts the game.
  - $\mathcal{B}_{II}$ randomly sample $\sigma^* \in Z_p$ and $C_0^* \in \{0,1\}^n$. $\mathcal{B}_{II}$ sets $C_1^* = xP_1$ and $C_2^* = cxP_1 + aH_1(\mathtt{ID}^*)xP_1$.
  - $\mathcal{B}_{II}$ returns $C^* = \langle C_0^*, C_1^*, C_2^*, \sigma^* \rangle$ as the challenge ciphertext.
- **Guess:** Once $\mathcal{A}_{II}$ outputs its guess b'. $\mathcal{B}_{II}$ chooses F_i from H_2^{list} or H_3^{list} and outputs K_i as the answer of the DH problem.

Now we analyse $\mathcal{B}_{II}$'s probability of outputting the correct answer to the DH problem. Similarly we define the following three events. **Event 1** is that $\mathcal{B}_{II}$ aborts the game prematurely. **Event 2** is that $H_2(*, xyP_1)$ or $H_3(*, xyP_1, *, *, *)$ is queried at some point during the simulation above. **Event 3** is that in the game $\mathcal{A}_{II}$ differentiates $\mathcal{B}_{II}$ from a real world before Event 2 happens.

Through a standard argument, we have

$$\Pr[\overline{\texttt{Event 1}}] \geq 1/q_1$$
$$\Pr[\texttt{Event 2}] \geq \epsilon(k)$$
$$\Pr[\overline{\texttt{Event 3}}] \geq (1 - 1/p)^{q_D}$$

Overall we have

$$\mathtt{Adv}_{\mathcal{B}_{II}}^{\mathtt{DH}_{1,1,1}}(k) \geq \frac{1}{q_2+q_3}\Pr[\overline{\mathtt{Event\ 1}} \wedge \mathtt{Event\ 2} \wedge \overline{\mathtt{Event\ 3}}] = \frac{\epsilon(k)}{q_P(q_2+q_3)}(1-1/p)^{q_D}.$$

□

The check $(\xi', C_1) \stackrel{?}{=} (v_0^{r'}, r'P_1)$ in the **CL.Decrypt** algorithm is important as the reduction implicitly requires this operation when responding to the $\mathbf{Decrypt}^S$ queries.

We mention that, if a general DH problem is given as $(P_1, xP_1, yP_1, P_2, xP_2)$, then $\mathcal{B}_{II}$ can make use of pairing as the decisional algorithm to test if $\hat{e}(F_i, P_2) = \hat{e}(yP_1, xP_2)$ to find the correct response to the DH problem. In that case, the reduction is tighter as

$$\mathtt{Adv}_{\mathcal{B}_{II}}^{\mathtt{General\ DH}}(k) \geq \Pr[\overline{\mathtt{Event\ 1}} \wedge \mathtt{Event\ 2} \wedge \overline{\mathtt{Event\ 3}}] = \frac{\epsilon(k)}{q_P}(1-1/p)^{q_D}.$$

4.7 CL-PKE3

Shi *et al.* [150] and Libert and Quisquater [118] presented two CL-PKEs referred to as SLOS-CL-PKE and LQ-CL-PKE respectively based on the same exponent inversion IBE: SK-IBE. Here for completeness we present another scheme CL-PKE3 based on SK-IBE, which can be faster than those two schemes when implemented with certain pairing parameters.

CL.Gen(1^k). On input 1^k, the algorithm works as follows.

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ of prime order p, an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$. Pick a random generator $P_2 \in \mathbb{G}_2^*$ and set $P_1 = \psi(P_2)$. Note that as SK-IBE, ψ is only required in the proof. In the implementation, one can randomly sample a generator P_1 from $\mathbb{G}_1$.

2. Pick a random $s \in \mathbb{Z}_p^*$ and compute $P_{pub} = sP_1$.

3. Pick four cryptographic hash functions:

$$H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*,$$
$$H_2 : \mathbb{G}_t \times \mathbb{G}_1 \rightarrow \{0,1\}^n,$$
$$H_3 : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p,$$
$$H_4 : \{0,1\}^n \rightarrow \{0,1\}^n,$$

for some integer $n > 0$.

4. Output the master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, H_1, H_2, H_3, H_4)$ and the master secret key $M_{\mathfrak{sk}} = s$.

The message space is $\mathbb{M} = \{0,1\}^n$, the ciphertext space is $\mathbb{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$ and the randomness space is $\mathbb{R} = \{0,1\}^n$.

CL.PartialKey$(M_{\mathfrak{sk}}, M_{\mathfrak{pk}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathfrak{pk}}$ and $M_{\mathfrak{sk}}$, the algorithm returns $D_A = \frac{1}{s+H_1(\mathtt{ID}_A)} P_2$.

CL.SecretVal$(M_{\mathfrak{pk}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm outputs a random $X_A \in \mathbb{Z}_p$.

CL.PrivateKey$(M_{\mathfrak{pk}}, D_A, X_A)$. Given $M_{\mathfrak{pk}}$, D_A and X_A, the algorithm outputs $S_A = (D_A, X_A)$.

CL.PublicKey$(M_{\mathfrak{pk}}, X_A, \mathtt{ID}_A)$. Given $M_{\mathfrak{pk}}$ and X_A, the algorithm computes $Q_A = H_1(\mathtt{ID}_A)P_1 + P_{pub}$ and outputs $P_A = X_A Q_A$.

CL.Encrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, m)$. Given a plaintext $m \in \{0,1\}^n$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathfrak{pk}}$ and the public key P_A of the entity, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, m, \mathtt{ID}_A, P_A)$.

2. Compute $Q_A = H_1(\mathtt{ID}_A)P_1 + P_{pub}$, $\xi = \hat{e}(P_1, P_2)^r$ and $f = rP_A$.

3. Set the ciphertext to $C = \langle rQ_A, \sigma \oplus H_2(\xi, f), m \oplus H_4(\sigma) \rangle$.

CL.Decrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle U, V, W \rangle \in \mathbb{C}$, the private key $S_A = (D_A, X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm takes the following steps:

1. Compute $\xi' = \hat{e}(U, D_A)$, $f' = X_A U$ and $\sigma' = V \oplus H_2(\xi', f')$.
2. Compute $m' = W \oplus H_4(\sigma')$ and $r' = H_3(\sigma', m', \mathtt{ID}_A, P_A)$.
3. Compute $Q_A = H_1(\mathtt{ID}_A)P_1 + P_{pub}$. If $U \neq r'Q_A$, output $\perp$, else return m' as the plaintext.

The security analysis can adopt the same strategy as in CL-PKE1 by first proving that the basic version is CL-OW-CPA secure and then applying the modified Fujisaki-Okamoto conversion. One can essentially follow the method in Theorem 3.5.3 to prove the basic version against the Type-I adversary, and one can prove the security against the Type-II adversary by following the method in Lemma 4.5.3.

Theorem 4.7.1. *CL-PKE3 is secure against Type-I adversary with CL-IND-CCA2 attacks provided H_i $(1 \leq i \leq 4)$ are modeled as random oracles and the ℓ-$BCAA1_{1,2}$ assumption is sound. CL-PKE1 is secure against Type-II adversary with CL-IND-CCA2 attacks provided H_i $(2 \leq i \leq 4)$ are modeled as random oracles and the $DH^{\psi}_{1,2,1}$ assumption is sound.*

Specifically, assume a Type-I adversary $\mathcal{A}_I$ breaks CL-PKE3 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$ and in the attack $\mathcal{A}_I$ makes q_D decryption queries and q_i queries on H_i for $1 \leq i \leq 4$ and $q_3 < 2^n$, then there exists an algorithm $\mathcal{B}_I$ to solve the $(q_1 - 1)$-$BCAA1_{1,2}$ problem with following advantage and time

$$\begin{aligned} Adv^{(q_1-1)-BCAA1_{1,2}}_{\mathcal{B}_I}(k) &\geq \tfrac{(1-q_2/2^n)\epsilon(k)}{q_1(q_3+q_4)}(1-\tfrac{1}{2^n})^{q_3 q_D}, \\ t_{\mathcal{B}_I}(k) &\leq t(k) + O(q_3 t_{\mathcal{E}} + q_2 \tau), \end{aligned}$$

where $t_{\mathcal{E}}$ is the cost of Basic-CL-PKE3 and τ is the time of a pairing.

Assume a Type-II adversary $\mathcal{A}_{II}$ breaks CL-PKE3 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$ and in the attack $\mathcal{A}_{II}$ makes q_P public key queries and q_i queries on H_i for $2 \leq i \leq 4$ and $q_3 < 2^n$, then there exists an algorithm $\mathcal{B}_{II}$ to solve the $DH^{\psi}_{1,2,1}$ problem with following advantage and time

$$\begin{aligned} Adv^{DH^{\psi}_{1,2,1}}_{\mathcal{B}_{II}}(k) &\geq \tfrac{\epsilon(k)}{q_P(q_3+q_4)}(1-\tfrac{1}{2^n})^{q_3 q_D}, \\ t_{\mathcal{B}_{II}}(k) &\leq t(k) + O(q_3 t_{\mathcal{E}} + q_2 \tau). \end{aligned}$$

In [118], Libert and Quisquater did not present the security analysis of their scheme. It appears that it is unlikely to prove the Libert-Quisquater scheme against the Type-II adversary based on the DH assumption. Shi *et al.* analysed their scheme's security against the Type-II adversary based on the BIDH assumption.

4.8 Efficiency Discussion and Comparison

We now assess the comparative efficiency of several concrete CL-PKE schemes. We only consider schemes with proofs in a security model compatible with the one defined in Section 4.2. As efficiency is the primary concern, we ignore any schemes designed in the standard model because their performance is worse than those with random oracles.

Table 4.4 summarises the used IBE scheme, the type of key construction, computation and ciphertext size of several CL-PKEs. The computation cost only counts the relevant basic operations including Pairing, Multiplication in $\mathbb{G}_1$ and Exponentiation in $\mathbb{G}_t$.

Table 4.4: CL-PKE Efficiency Comparison

Schemes	Based IBE	Type of Key Construction	Computation	Ciphertext Size
AP-CL-PKE1 [5](1)	BF-IBE	Full Domain Hash	=BF-IBE+$2P$(2)	=BF-IBE
AP-CL-PKE2 [6](3)	BF-IBE	Full Domain Hash	=BF-IBE+$1M$	=BF-IBE
CL-PKE1	BF-IBE	Full Domain Hash	=BF-IBE+$1M$	=BF-IBE
CL-PKE2	BB_1-IBE	Commutative Blinding	=BB_1-IBE+$1M$	=BB_1-IBE
LQ-CL-PKE [118]	SK-IBE	Exponent Inversion	=SK-IBE+$1E$	=SK-IBE
SLOS-CL-PKE [150]	SK-IBE	Exponent Inversion	=SK-IBE+$1M$(4)	=SK-IBE
CL-PKE3	SK-IBE	Exponent Inversion	=SK-IBE+$1M$	=SK-IBE

1. The scheme is enhanced by a verifiable random parameter generation algorithm.
2. The scheme requires two more pairings in the encryption algorithm than BF-IBE. The decryption operation is of the same cost as BF-IBE.
3. The scheme is enhanced as suggested in Section 4.5.4.
4. The scheme requires the Weil pairing.

Among the schemes using the full domain hash IBEs, CL-PKE1 and AP-CL-PKE2 have same performance and are faster than AP-CL-PKE1 in encryption. Note that AP-CL-PKE2

employs the "exclusive-or" structure which make it suitable for security-mediated encryption where a security mediator can complete partial decryption [44]. Among the schemes using the exponent inversion IBEs, LQ-CL-PKE and CL-PKE3 have similar computation cost. CL-PKE3 can be slightly faster than LQ-CL-PKE with certain implementations where the multiplication in $\mathbb{G}_1$ is faster than the exponentiation in $\mathbb{G}_t$, for example, using the Type-3 pairing with the 128-bit security, the multiplication in $\mathbb{G}_1$ is three times faster than the exponentiation in $\mathbb{G}_t$ according to Table 3.8. CL-PKE3 also has shorter public keys than LQ-CL-PKE. SLOS-CL-PKE makes use of the Weil pairing. As the Weil pairing is slower than the Tate pairing [87], SLOS-CL-PKE is slower than LQ-CL-PKE and CL-PKE3. Like BB_1-IBE, CL-PKE2 enjoys security reductions based on weak complexity assumptions and is efficient in encryption and relatively slow in decryption. CL-PKE3 and LQ-CL-PKE can be easily modified to use the "exclusive-or" structure for security mediation.

4.9 Conclusion

In this chapter, we revisited various CL-PKE formulations and defined a strong security notion for this type of encryption. By making a simple observation on some existing IBEs and PKEs, we then proposed a heuristic approach to constructing efficient CL-PKEs from IBEs. Following the approach, we presented three robust and efficient concrete CL-PKEs using three types of IBE. As the schemes constructed from IBEs, they can smoothly degenerate to an IBE. Beside, we also demonstrated that a slightly modified Fujisaki-Okamoto conversion can transform a weak CL-PKE to a CL-IND-CCA2 secure scheme.

Chapter 5

Identity-Based Key Agreement Protocols

In this chapter we investigate another important cryptographic primitive: key agreement protocols; in particular, we focus on the identity-based two-party key agreement protocols constructed with pairings. We first enhanced the Bellare-Rogaway key agreement model to cover all the commonly required security properties. Then we formally analyse several existing protocols including the enhanced Smart protocol, the enhanced Shim protocol and the McCallugh-Barreto protocols in the enhanced model. The formal security analysis can build confidence in the security of these schemes. Finally we present an identity-based key agreement protocol with unilateral identity privacy which is suitable for certain environments where identity privacy is important.

5.1 Introduction

Key Agreement Protocols (KAP) are the mechanisms by which two or more parties can establish an agreed secret key over a network fully controlled by adversaries. Normally the established key should vary on each execution (session) of the protocol. If in a protocol one party is assured that no other party aside from the specifically identified party (or parties) may gain access to the particular established secret key, then the key agreement protocol

is said to provide (implicit) key authentication. A key agreement protocol which provides mutual (implicit) key authentication between (or among) parties is called an *Authenticated Key agreement* (AK). Although an AK provides key authentication, one party is not sure whether the other party (or other parties) actually has possession of the established secret; otherwise, the protocol is said to provide key confirmation. If a key agreement protocol provides both key authentication and key confirmation, it is called an *Authenticated Key agreement with key Confirmation* (AKC) [127].

A number of security properties are generally believed to be necessary (or good) for an AK and AKC.

1. *Known session key security (KSK).* Each execution of the protocol should result in a unique secret session key. The compromise of one session key should not help compromise the keys established in other sessions, e.g. parallel sessions, previous sessions and future sessions.

2. *Forward secrecy (FS).* If the long-term keys of one or more parties are compromised, the secrecy of previously established session keys should not be affected. We say that a protocol has *partial forward secrecy: s-FS*, if one or more but not all of the parties' long-term keys can be corrupted without affecting previously established session keys between these parties, and we say that a protocol has *perfect forward secrecy: p-FS*, if the long-term keys of all the parties involved may be corrupted without affecting the security of any session key previously established between these parties. In the identity-based cryptosystems, there is a stronger notion of forward secrecy, called *master key forward secrecy: m-FS*. Specifically, the KGC's long-term secret key may be corrupted and so are all the parties' long-term private keys, but the security of previously established session keys by any parties should not be affected.

3. *Key-compromise impersonation resilience (KCI).* The compromise of party A's long-term private key will allow an adversary to impersonate A, but it should not enable the adversary to impersonate other parties to A.

4. *Unknown key-share resilience (UKS).* Party A should not be able to be coerced into sharing a key with party C when in fact A thinks that it is sharing the key with party B.

Authenticated key agreement protocols can be built from both symmetric-key security mechanisms such as secret-key cipher and message authentication code and asymmetric-key security mechanisms such as public key encryption and signature. If the symmetric-key mechanism is used, then either a random symmetric key or a human-chosen password should be distributed before the key agreement protocol is executed. Without the involvement of a trusted third party such as a key distribution center or a key transport center, the number of the shared symmetric keys or passwords grows dramatically with the number of users. While, with a trusted party in the system, the trusted party has to be online to participate in each protocol execution. This complicates the protocols and also generates a single point of failure. If the traditional (certificate-based) public key cryptography is used, an infrastructure such as PKI to manage certificates should be deployed. It has been proven that such systems are difficult to build and manage. Another solution is to use the identity-based cryptography (IBC) [138] in which each party's public key is the party's identity. Hence, no certificate is required in IBC and the management of public keys is greatly simplified.

Two party identity-based key agreement protocols (IB-KAP) constructed from pairings are our focus in this chapter. We will formally analyse a few existing two-party identity-based authenticated key agreements from pairings including the enhanced Smart

protocol [143, 63], the enhanced Shim protocol [144, 166] and the McCallugh-Barreto protocols [124, 125]. These security analyses will be conducted in a properly defined two-party key agreement security model originated from Bellare and Rogaway's seminal work [32]. Then we present an IB-KAP with unilateral identity privacy. The protocol apart from establishing authenticated secret also hides a user's identity from an adversary, and is particularly suitable for certain environments where identity privacy is important.

To convert an AK to an AKC, a well-known method is to make use of a secure message authentication code (MAC) algorithm, which as defined in [15] takes a key and a message as input and outputs a tag on the message. A and B each generate such a tag by using the shared MAC key derived from the agreed secret by a key derivation function and a selected message that could include A and B's identifiers, protocol transcript and a message index. A and B then exchange their tags to confirm that they have got the shared session secret [23, 59, 116]. In the following part of the chapter we just focus on the security of AKs.

5.2 Two-Party Key Agreement Security Model

Key agreement protocol design has a long history. However, many protocols were designed in *ad hoc* modes. Proposed protocols were often broken, sometimes years after they were first presented. To address this problem, some methodologies have been developed to check or prove the security of protocols.

In 1993, using the indistinguishability notion which requires that an adversary cannot in polynomial time differentiate the agreed key from a random sample generated from the agreed key's distribution, Bellare and Rogaway [32] formalised a model (from now on, we will refer to it as the Bellare-Rogaway model) of symmetric key based two-party authenticated key agreements by borrowing the notion "matching protocol runs" from [71]. Based on the Bellare-Rogaway model, some extensions were made to formalise other types

of authenticated key agreement protocol, including the two-party asymmetric key based KAP [38], two-party KAP with a trusted server [34], KAP with extra security properties, e.g. forward secrecy [30], anti-dictionary attack [30], smart card model [154] and AK with confirmation [23]. More complicated cases, the group KAP models, can be found in [17, 18, 135].

In 1998, Bellare *et al.* proposed another KAP model [16] using the simulation paradigm. Later, Shoup [146] presented a different model based on the simulation strategy. Canetti and Krawczyk readdressed Bellare *et al.*'s simulation model and proposed the secure channel notion in [59]. A more general paradigm, the so-called universally composable security, can be found in [60].

A different line of work, using a "logic-based" approach to "reason" that an authentication protocol is correct, was pioneered by Burrows, Abadi and Needham [8]. Although this idea is useful and easy to apply, it has a serious drawback: a proof of the correctness of a protocol does not guarantee that the protocol is secure. Abadi and Rogaway tried to combine the advantages of "logic reasoning" and "provable indistinguishability model" in a unified approach [7].

In this chapter we will use a modified Bellare-Rogaway key exchange model [32] to analyse a number of two-party protocols. In the model, each party participating in a session is treated as an oracle. An oracle $\Pi^s_{i,j}$ denotes the s-th instance of party i involved with a partner party j in a session. Note that an oracle $\Pi^s_{i,j}$ can be uniquely identified by i and s. The oracle $\Pi^s_{i,j}$ executes the prescribed protocol Π and produces the output as $\Pi(1^k, i, j, S_i, P_i, P_j, tran^s_{i,j}, r^s_{i,j}, x) = (m, \delta^s_{i,j}, \sigma^s_{i,j}, j)$ where x is the input message; m is the outgoing message; S_i and P_i are the private/public key pair of party i; $\delta^s_{i,j}$ is the decision of the oracle (accept/reject the session or no decision yet); $\sigma^s_{i,j}$ is the generated session key and P_j is the public key of the intended partner j (see [32, 23] for more details). After the response is generated, the conversation transcript $tran^s_{i,j}$ is updated as $tran^s_{i,j}.x.m$, where

"$a.b$" denotes the result of the concatenation of two strings, a and b. An adversary can access an oracle by issuing some specified queries defined in the game below.

The security of a protocol is defined through a two-phase game between an adversary $\mathcal{A}$ and a simulator which simulates the executions of a protocol by providing the adversary with access to oracles. In the first phase, $\mathcal{A}$ is allowed to issue the following queries to oracles in any order.

1. $Send(\Pi^s_{i,j}, x)$. Upon receiving the message x, oracle $\Pi^s_{i,j}$ executes the protocol and responds with an outgoing message m or a decision to indicate accepting or rejecting the session. If the oracle $\Pi^s_{i,j}$ does not exist, it will be created as an initiator, the party who sends out the first message in the protocol, if $x = \lambda$, or as a responder otherwise. In this thesis, we require $i \neq j$, namely, a party will not run a session with itself. Such restriction is not unusual in practice.

2. $Reveal(\Pi^s_{i,j})$. If the oracle has not accepted, it returns $\perp$; otherwise, it reveals the session key.

3. $Corrupt(i)$. The party i responds with its private key.

Once the adversary decides that the first phase is over, it starts the second phase by choosing a *fresh oracle* $\Pi^s_{i,j}$ and issuing a $Test(\Pi^s_{i,j})$ query, where the *fresh oracle* $\Pi^s_{i,j}$ and $Test(\Pi^s_{i,j})$ query are defined as follows.

Definition 5.2.1. (fresh oracle) An oracle $\Pi^s_{i,j}$ is fresh if (1) $\Pi^s_{i,j}$ has accepted; (2) $\Pi^s_{i,j}$ is unopened (not been issued the *Reveal* query); (3) party $j \neq i$ is not corrupted (not been issued the *Corrupt* query); (4) there is no opened oracle $\Pi^t_{j,i}$, which has had a matching conversation to $\Pi^s_{i,j}$.

The above fresh oracle is particularly defined to cover the key-compromise impersonation resilience property since it implies that party i could have been issued a *Corrupt* query.

4. $Test(\Pi^s_{i,j})$. Oracle $\Pi^s_{i,j}$ which is fresh, as a challenger, randomly chooses $b \in \{0, 1\}$

and responds with the session key if $b = 0$, or a random sample from the distribution of the session key otherwise.

After this point the adversary can continue querying the oracles except that it cannot reveal the test oracle $\Pi^s_{i,j}$ or an oracle $\Pi^t_{j,i}$ which has a matching conversation to $\Pi^s_{i,j}$ if such an oracle exists, and it cannot corrupt party j. Finally the adversary outputs a guess b' for b. If $b' = b$, we say that the adversary wins. The adversary's advantage is defined as

$$Adv^{\mathcal{A}}(k) = |2\Pr[b' = b] - 1|.$$

We use session ID to define *matching conversations*. Two oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ have a matching conversation to each other if both of them have the same session ID. In this thesis, we will use the concatenation of the messages in a session (the transcript of an oracle) to define the session ID.

A secure authenticated key (AK) agreement protocol is defined as follows.

Definition 5.2.2. Protocol Π is a secure AK if:

1. In the presence of a benign adversary, which faithfully conveys messages, on $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$, both oracles always accept holding the same session key, and this key is distributed uniformly in the session key space;
2. For any polynomial time adversary $\mathcal{A}$, $\mathtt{Adv}^{\mathcal{A}}(k)$ is a negligible function of security parameter k.

It is straightforward to see that when a party will not run a session with itself, if a protocol is secure regarding Definition 5.2.1 and 5.2.2, then it is secure in a weaker security model in which the fresh oracle $\Pi^t_{i,j}$ requires that both party i and j are uncorrupted (such fresh oracle is used in [32, 23]).

If a protocol is secure regarding the above formulation, it achieves implicit mutual key authentication and the following general security properties: the known session key security, the key-compromise impersonation resilience and the unknown key-share resilience [23, 64].

- Mutual key authentication. Condition 2 in Definition 5.2.1 guarantees that the session key can only be computed by the engaged two parties. Therefore any protocol that is secure under the definition provides the mutual key authentication.

- Known session key security. The *Reveal* query allows the adversary to compromise session keys. So Condition 2 in Definition 5.2.1 guarantees that adversary is not able to learn any information of the chosen session key even if the adversary may have revealed other session keys apart from the information it can learn without revealing those keys.

- Key-compromise impersonation resilience. This is an essential property of an entity authentication mechanism. For an AK protocol, we slightly change the notion as that the compromise of party A's long-term private key should not enable the adversary to impersonate other parties to A in a session by computing any bit information of the session key established with A. The fresh oracle definition is particularly designed to capture the notion that an adversary may have compromised a party's long-term private key. Condition 2 in Definition 5.2.1 therefore guarantees that the adversary still cannot learn any bit of information of the session key established by party A in a session with an uncorrupted party which may be impersonated by the adversary, even if the adversary has compromised A's long-term key.

- Unknown key-share resilience. Suppose in a protocol Π, a party i is coerced into sharing a session key SK in a protocol execution s with a party j in an execution t, but i believes sharing the key with party w in the s-th execution of Π. The protocol can be easily defeated by an adversary with the allowed queries in the model as follows: the adversary reveals the oracle $\Pi^s_{i,w}$ to get key SK and chooses $\Pi^t_{j,i}$ as the fresh oracle in the *Test* query and the adversary wins the game trivially. Hence the unknown key-share resilience is guaranteed if a protocol is secure regarding Definition 5.2.1.

Now we consider the forward secrecy property. Informally, forward secrecy of a protocol requires that the security of a session key established by a party is not affected even if the long-term key of either the party or its peer party in the session is compromised. We emphasize that the peer party in the session may be impersonated by the adversary. Many AK protocols (with implicit key authentication) including the protocols presented in this chapter cannot achieve forward secrecy in this sense. Specifically, an adversary $\mathcal{A}$ can breach the security as follows: $\mathcal{A}$ first impersonates party A to B in a session by choosing an ephemeral secret and generating corresponding messages following the protocol specification. After B receives all the messages and completes the session by computing the session key, $\mathcal{A}$ then corrupts A and computes the session key established by B with the knowledge of the used ephemeral secret and A's long-term key.

Because it appears there is no natural way to identify that the adversary does not know the ephemeral secret of a message in the test session, we use a slightly "weaker" notion, which requires the adversary to be benign in the test session and with the knowledge of one party's long-term key to distinguish the session key from a random sample to win the game. This appears able to reflect the common requirement on a protocol with forward secrecy. Moveover, we give the adversary an extra capacity that it can corrupt a party at any time in the game.

Definition 5.2.3. An AK protocol is said to be forward secure if any polynomial time adversary wins the game with negligible advantage when it chooses as the challenger (i.e. in place of the fresh oracle) an *unopened* oracle $\Pi^s_{i,j}$ which has a matching conversation to another *unopened* oracle $\Pi^t_{j,i}$ and both oracles accepted and only one of i and j can be corrupted. If both i and j can be corrupted, then the protocol achieves perfect forward secrecy. If in the game, the master secret key can be disclosed, then the protocol achieves master key forward secrecy. The corruption of long-term keys or the disclosure of the master secret key may happen at any time of the game.

Regarding the master key forward secrecy, the adversary may be a passive-but-malicious KGC as in CL-PKE in Section 4.2. Hence in a stronger formulation, the adversary should be allowed to setup the system parameters.

5.3 Review on Existing Schemes from Pairing

After Shamir proposed the concept of IBC, a number of IB-KAP schemes were proposed based on RSA or DL problem, e.g. [100, 133, 158].

In 2000, Sakai *et al.* introduced an identity-based non-interactive key establishment scheme based on bilinear pairings over elliptic curves [152]. In this work, they proposed an important ID key construction from pairing: the SOK key construction (see Section 3.3 for more information on ID key constructions). After that, many other IB-KAPs using this key construction were presented, such as the Smart protocol [143], Scott protocol [137], CK protocol [63], Shim protocol [144], BMP protocol [26] and Wang protocol [161]. Among those proposals, the Smart protocol [143] is the first two-party key agreement standing firmly against heuristical security analysis. As the Smart protocol does not achieve PFS and master-FS, Chen and Kudla proposed an enhanced variant of the scheme, which we refer to as the Smart-Chen-Kudla scheme (SCK for short) [63]. While, the Smart protocol requires two pairing operations which seem costly[1], Shim then proposed a protocol using a single pairing in [144]. However, the Shim protocol suffers from the man-in-the-middle attack [149] (see the attack in Section 5.4). Later, as observed by Yuan and Li [166], the Shim protocol can be easily enhanced to recover from the attack. Moreover, Yuan and Li heuristically argued that the enhanced protocol, which we refer to as the Shim-Yuan-Li protocol (SYL for short), exhibits many desired security properties.

In 2003, Sakai and Kasahara presented a new identity-based key construction using pairings, i.e. the SK key construction [151] (see Section 3.3 for details). Based on the SK key construction, McCullagh and Barreto presented an identity-based authenticated key agreement protocol (referred to as MB-1) on CT-RSA 2005 [125], which appears to be more efficient in computation than those based on the SOK key construction. However, as we

[1] With the progress on understanding pairings, it has been shown that the Smart protocol is in fact more efficient than many other protocols on certain types of pairing [52].

pointed out [124], the scheme is vulnerable to a key-compromise impersonation attack (see the attack in Section 5.5). In order to recover from this security weakness, McCullagh and Barreto [124], and Xie [163] proposed two fixes (called MB-2 and Xie protocol) respectively. Unfortunately, Xie's protocol still suffers from the key-compromise impersonation attack [119] (see the attack in Section 5.5). Li *et al.* proposed some other variants of the MB protocols with some heuristical security analysis [119].

As demonstrated by numerous cases in the literature, such as the attack on the Shim protocol, the key agreements designed in *ad hoc* modes are often vulnerable to certain attacks. Some formal analysis, normally a valid reduction based on a hardness assumption in a formal security model, is desirable for one to build confidence in the security of schemes. Among these protocols, some of them have been formally analysed in the Bellare-Rogaway model based on various assumptions, such as the Smart protocol which was formally analysed in [109] based on the GBDH assumption, CK protocol which was analysed in a weaker variant of the Bellare-Rogaway model based on the BDH assumption, BMP protocol which is analysed based on the BDH assumption and Wang protocol which is analysed based on the DBDH assumption. Some other protocols such as the MB protocols and Xie protocol, have a security analysis but the security reduction is invalid.

The properties of those protocols, including the security attributes, existence of reduction, used key construction, implementable pairings and computation complexity, are summarised in Table 5.1.

In the first part of this chapter we will formally analyse three protocols including SCK, SYL and MB-2 in the modified Bellare-Rogaway model presented in Section 5.2. The SCK protocol exhibits very strong security properties. Moreover it could be implemented with pairings on the ordinary curves to achieve the best performance among those protocols that use the SOK key construction and have the same security strength, even though the protocol requires two pairing operations (see [52] for the detailed performance comparison). We will

Table 5.1: Identity-based Key Agreements from Pairings

Schemes	Key Const.	Pairing Type	Security Properties						Reduction	Computation
			KSK	FS			KCI	UKS		
				s	p	m				
Smart[143]	SOK	1,2,3,4	✓	✓	×	×	✓	✓	GBDH	$2P + 2M$
SCK[60]	SOK	1,2,3,4	✓	✓	✓	✓	✓	✓	BDH[(1)]	$2P + 3M$
CJL[58]	SOK	1,2,3,4	✓	✓	✓	✓	✓	✓	-	$2P + 4M$
Shim[144]	SOK	1, 4	b	r	o	k	e	n	-	$1P + 2M$
SYL[166]	SOK	1, 4	✓	✓	✓	✓	✓	✓	BDH[(2)]	$1P + 3M$
BMP[26]	SOK	1, 4	✓	✓	✓	✓	×	✓	BDH	$1P + 2M$
Scott[137]	SOK	1, 4	-	✓	✓	✓	×	-	-	$1P + 2E$
CK[63]	SOK	1, 4	✓	✓	×	×	✓	✓	GBDH	$1P + 2M$
Wang[161]	SOK	1, 4	✓	✓	✓	×	✓	✓	DBDH	$1P + 3M$
MB-1[125]	SK	1,2,3,4	-	✓	✓	×	×	-	-	$1P + 3M$
MB-2[124]	SK	1,2,3,4	✓	✓	×	×	✓	✓	ℓ-GBCAA1[(3)]	$1P + 1E + 2M$
Xie[163]	SK	1,2,3,4	b	r	o	k	e	n	-	$1P + 1E + 2M$
LYL-1[119]	SK	1,2,3,4	-	-	-	×	-	-	-	$1P + 2E + 2M$
LYL-2[119]	SK	1,2,3,4	-	-	-	×	-	-	-	$1P + 2E + 2M$

1. The scheme is proven in Section 5.4.1.
2. The scheme is proven in Section 5.4.2.
3. The scheme is proven in Section 5.5.3.

formally analyse the protocol in Section 5.4.1 based on the weakest possible BDH assumption for this type of protocol. The SYL protocol achieves the same security properties as the SCK protocol and can have better performance if it is implemented with the Type-1 pairings on the supersingular curves. We will formally analyse the SYL protocol in Section 5.4.2 based on the BDH assumption. As a result of using the SK key construction, the MB protocols are the most efficient ones among all the existing schemes. We will point out the errors in the existing security analysis and formally analyse a simple variant of the MB-2 protocol in Section 5.5 but based on the stronger ℓ-GBCCA1 assumption.

5.4 Security Analysis of the SCK and SYL Protocol

In this section we formally analyse the security of two schemes based on the SOK key construction. The first one is the Smart-Chen-Kudla (SCK) scheme in [60] and the second one is the Shim-Yuan-Li (SYL) scheme in [166]. SCK is the most robust and efficient scheme

with pairings on the ordinary curves (the Type-2, 3, 4 pairings) for the high security level such as the 128-bit security level. On the other hand, SYL is the strongest and the most efficient scheme with the pairings on the supersingular curves (the Type-1 pairings) for the relative low security level such as the 80-bit security level. The two protocols use the same key construction and exchange same message flows as follows:

Setup. The KGC executes the following operations:

1. On input 1^k, generate a set of pairing parameters of the required size.

2. Pick a random $s \in \mathbb{Z}_p$ and compute $R = sP_2$.

3. Pick two cryptographic hash functions as follows:

$$H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1(\text{resp. } \mathbb{G}_2),$$
$$H_2 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_2 \times \mathbb{G}_2 \times \mathbb{G}_2 \times \mathbb{G}_t \rightarrow \{0,1\}^n$$

   for some integer $n > 0$. The codomain of H_1 is $\mathbb{G}_1$ for the SCK protocol and $\mathbb{G}_2$ for the SYL protocol.

4. Set s as the master secret key which is kept secret by KGC and publish others as the system public parameters (also called the master public key).

Extract. For any user with an identity $\mathtt{ID}_A \in \{0,1\}^*$, given the master secret key s and the system parameters, the KGC executes the SOK **Extract** algorithm to compute $Q_A = H_1(\mathtt{ID}_A)$, $D_A = sQ_A$ and passes D_A as the private key to this user via some secure channel.

In the protocols, party A and B randomly choose x and y from $\mathbb{Z}_p$ respectively and perform the protocol as follows:

$$A \rightarrow B : T_A = xP_2$$
$$B \rightarrow A : T_B = yP_2$$

On completion of the protocol, A and B use one of following schemes to compute a session secret shared between them.

The SCK scheme: A computes $K = \hat{e}(xQ_B, R) \cdot \hat{e}(D_A, T_B)$ and $xT_B = xyP_2$, and B computes $K = \hat{e}(yQ_A, R) \cdot \hat{e}(D_B, T_A)$ and $yT_A = xyP_2$. The session key is computed by $SK = H_2(A, B, T_A, T_B, xyP_2, K)$.

The SYL scheme: A computes $K = \hat{e}(\psi(T_B + Q_B), xR + D_A)$ and $xT_B = xyP_2$, and B computes $K = \hat{e}(\psi(yR + D_B), T_A + Q_A)$ and $yT_A = xyP_2$. The session key is computed as $SK = H_2(A, B, T_A, T_B, xyP_2, K)$.

Both the original Smart [143] and Shim [144] protocol do not include the DH value xyP_2 in the session key generation function by the hash function H_2 and both have vulnerabilities. The Smart protocol does not achieve p-FS and m-FS. With the knowledge of $D_A = sQ_A$ and $D_B = sQ_B$, the session key can be computed as $SK = H_2'(A, B, T_A, T_B, \hat{e}(D_A, T_B) \cdot \hat{e}(D_B, T_A))$. The Shim protocol suffers from a man-in-the-middle attack as follows [149]: Adversary $\mathcal{A}$ randomly chooses $x', y' \in \mathbb{Z}_p$, and intercepts and replaces A and B's messages

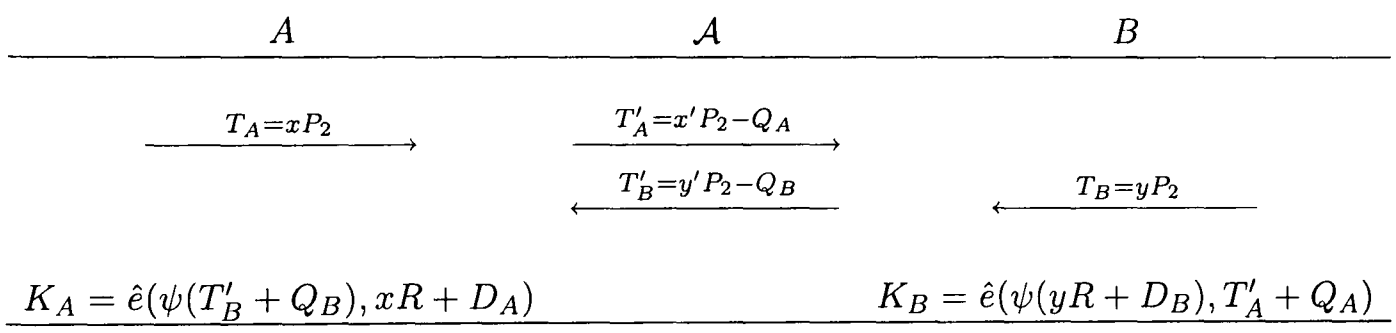

Figure 5.1: Man-In-The-Middle Attack on The Shim Protocol

with $x'P_2 - Q_A$ and $y'P_2 - Q_B$ respectively as in Figure 5.1. $\mathcal{A}$ can compute both secrets $K_A = \hat{e}(\psi(T_A), R)^{y'} \cdot \hat{e}(\psi(R), Q_A)^{y'}$ and $K_B = \hat{e}(\psi(R), T_B)^{x'} \cdot \hat{e}(\psi(Q_A), R)^{x'}$ established by A and B respectively.

By including the DH value xyP_2 in the session key generation function, the two protocols become more robust and can achieve all the security properties presented in Section 5.1. In

the following part of this section, we formally analyse the security of these two protocols.

5.4.1 Security Analysis of the SCK Protocol

In this subsection, we formally analyse the SCK scheme. The security of the scheme can be summarised by Theorem 5.4.1 and 5.4.2.

Theorem 5.4.1. *The SCK scheme is a secure AK, provided the $BDH^{\psi}_{2,1,2}$ assumption holds and the hash functions are modeled as random oracles. Specifically, suppose in the attack, a PPT adversary $\mathcal{A}$ which makes q_i queries to H_i for $i = 1, 2$ and creates q_o oracles, wins the game with non-negligible advantage $\epsilon(k)$. Then there exists a PPT algorithm $\mathcal{B}$ to solve the $BDH^{\psi}_{2,1,2}$ problem with advantage*

$$Adv_{\mathcal{B}}^{BDH^{\psi}_{2,1,2}}(k) \geq \frac{1}{q_1 \cdot q_o \cdot q_2}\epsilon(k).$$

Notice, that the proof is relative to an oracle which computes the isomorphism. In pairing parameter instances where such an isomorphism exists, this is equivalent to the $\mathrm{BDH}_{2,1,2}$ problem. However, one can implement the SCK scheme for pairing parameters which do not have an explicitly computable isomorphism.

Proof: We define the session ID as the concatenation of $xP_2 \| yP_2$. The first condition in Definition 5.2.2 is trivial to prove. Now we prove that the protocol meets the second condition.

Given a $\mathrm{BDH}^{\psi}_{2,1,2}$ problem instance (aP_2, bP_1, cP_2), we construct an algorithm $\mathcal{B}$ using the adversary $\mathcal{A}$ against the protocol to solve the $\mathrm{BDH}^{\psi}_{2,1,2}$ problem.

$\mathcal{B}$ simulates the system setup to adversary $\mathcal{A}$ as follows. The system public parameters includes the pairing parameters of the input problem, $R = aP_2$ (hence $\mathcal{B}$ does not know the master secret key), and two functions H_1 and H_2 which are instantiated as random oracles under the control of $\mathcal{B}$.

Algorithm $\mathcal{B}$ randomly chooses $1 \leq I \leq q_1$ and $1 \leq J \leq q_o$ and starts simulating the real world where the adversary $\mathcal{A}$ launches the attack. Algorithm $\mathcal{B}$ answers the following queries, which are asked by adversary $\mathcal{A}$ in an arbitrary order. We slightly abuse the notation $\Pi^t_{i,j}$ to refer to the t-th party instance among all the party instances created in the attack, instead of the t-th instance of party i. This would not affect the soundness of the security model.

- $H_1(\mathtt{ID}_i)$: Algorithm $\mathcal{B}$ maintains an initially empty list H_1^{list} with entries of the form $(\mathtt{ID}_i, Q_i, \ell_i)$. The algorithm $\mathcal{B}$ responds to the query in the following way.
  - If $\mathtt{ID}_i$ already appears on H_1^{list} in a tuple $(\mathtt{ID}_i, Q_i, \ell_i)$, then $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = Q_i$.

- Otherwise, if $\mathtt{ID}_i$ is the I-th unique identifier query, then $\mathcal{B}$ inserts $(\mathtt{ID}_i, bP_1, \perp)$ into the list and returns bP_1 (hence the private key of $\mathtt{ID}_I$ should be abP_1 which is not known by the algorithm $\mathcal{B}$).
- Otherwise, $\mathcal{B}$ randomly chooses $\ell_i \in \mathbb{Z}_p$, inserts $(\mathtt{ID}_i, \ell_i P_1, \ell_i)$ into the list and returns $\ell_i P_1$.

- $H_2(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u)$: Algorithm $\mathcal{B}$ maintains an initially empty list H_2^{list} with entries of the form $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u, h_u)$. The algorithm $\mathcal{B}$ responds to the query in the following way (for easily following the reduction, we suggest one reading the Send and Reveal query first).
  - If a tuple indexed by $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u)$ is on the list, then $\mathcal{B}$ responds with h_u.
  - Otherwise, $\mathcal{B}$ goes through the list $\mathcal{L}$ (maintained in the Reveal query) to find a tuple with values $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, \Pi_{i,j}^t)$ and proceeds as follows (in the following, without loosing generality, we assume Y_u is the message generated by oracle $\Pi_{i,j}^t$, so X_u is the incoming message to $\Pi_{i,j}^t$. By the behavior of Send query, $Y_u = f_{i,j}^t aP_2$).
    * Test if $\hat{e}(\psi(Y_u), X_u) = \hat{e}(P_1, Z_u)$ holds and for Type 2 or 4 pairings test if $\hat{e}(\psi(X_u), Y_u) = \hat{e}(\psi(Z_u), P_2)$ holds as well. If the equations hold, $\mathcal{B}$ then does the following:
      · Find the values $f_{i,j}^t$ and $SK_{i,j}^t$ corresponding to oracle $\Pi_{i,j}^t$ from the list Ω maintained in the Send query.
      · Find the value ℓ_j from H_1^{list} for party with $\mathtt{ID}_j$.
      · Compute the shared secret via the following equation

$$\begin{aligned} K_{i,j}^t &= \hat{e}(x_i Q_j, R) \cdot \hat{e}(D_i, X_u) \\ &= \hat{e}(f_{i,j}^t a \ell_j P_1, aP_2) \cdot \hat{e}(abP_1, X_u), \\ &\quad \text{since } x_i = f_{i,j}^t a, Q_j = \ell_j P_1, D_i = abP_1 \\ &= \hat{e}(f_{i,j}^t \ell_j \psi(aP_2), aP_2) \cdot \hat{e}(bP_1, \frac{1}{f_{i,j}^t} Z_u), \text{ since } f_{i,j}^t a X_u = Z_u. \end{aligned}$$

      Note that $\Pi_{i,j}^t$ is put on the list $\mathcal{L}$ in the Reveal query only when $\Pi_{i,j}^t$ has been revealed and $D_i = abP_1$, but $H_2(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_{i,j}^t)$ had not been queried before the Reveal query. So, $SK_{i,j}^t$ has been randomly sampled.
      · Set $h_u = SK_{i,j}^t$.
      · Remove $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, \Pi_{i,j}^t)$ from the list $\mathcal{L}$. Put $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_{i,j}^t, h_u)$ in the list H_2^{list}.
      · Check if $K_{i,j}^t = K_u$. If it is not true, $\mathcal{B}$ randomly chooses new $h_u \in \{0,1\}^n$, inserts $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u, h_u)$ into the list H_2^{list}.

· Return h_u.

* Otherwise (no tuple on $\mathcal{L}$ meets the test), algorithm $\mathcal{B}$ randomly chooses $h_u \in \{0,1\}^n$, inserts $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u, h_u)$ into the list and returns h_u.

– Otherwise (no related tuple on $\mathcal{L}$ is found), $\mathcal{B}$ randomly chooses $h_u \in \{0,1\}^n$, inserts $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u, h_u)$ into the list and returns h_u.

- **Corrupt**($\mathtt{ID}_i$): $\mathcal{B}$ looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. $\mathcal{B}$ checks the value of ℓ_i: if $\ell_i \neq \perp$, then $\mathcal{B}$ responds with $\ell_i \psi(aP_2) = \ell_i aP_1$; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).

- **Send**($\Pi_{i,j}^t$, **M**): $\mathcal{B}$ maintains a list Ω for each oracle of the form $(\Pi_{i,j}^t, tran_{i,j}^t, r_{i,j}^t, K_{i,j}^t, SK_{i,j}^t, f_{i,j}^t)$ where $tran_{i,j}^t$ is the transcript of the oracle so far; $r_{i,j}^t$ is the random integer used by the oracle to generate message, $f_{i,j}^t$ is used for special purpose explained below, and $K_{i,j}^t$ and $SK_{i,j}^t$ are set $\perp$ initially. Note that the Ω list can be updated in other queries as well, such as the Reveal query and the H_2 query. $\mathcal{B}$ proceeds in the following way:

  – If M is the second message on the transcript, do nothing but simply accept the session. Otherwise,

  – Query $H_1(\mathtt{ID}_i)$ and $H_1(\mathtt{ID}_j)$.

  – If $t = J$,

    * If $\ell_j \neq \perp$, then abort the game (**Event 2**).

    * Otherwise, respond with cP_2 and set $r_{i,j}^t = \perp$ (if $M = \lambda$, then party $\mathtt{ID}_i$ is an initiator, otherwise a responder as M is the first message of the session).

  – Otherwise,

    * If $\ell_i = \perp$, randomly choose $f_{i,j}^t \in \mathbb{Z}_p$ and respond with $f_{i,j}^t aP_2$ and set $r_{i,j}^t = \perp$.

    * Otherwise, randomly choose $r_{i,j}^t \in \mathbb{Z}_p$ and respond with $r_{i,j}^t P_2$.

- **Reveal**($\Pi_{i,j}^t$): $\mathcal{B}$ maintains a list $\mathcal{L}$ with tuples of the form $(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, \Pi_{i,j}^t)$, $\mathcal{B}$ proceeds in the following way to respond:

  – Get the tuple of oracle $\Pi_{i,j}^t$ from Ω.

  – If oracle $\Pi_{i,j}^t$ has not accepted, then respond with $\perp$.

  – If $t = J$ or if the J-th oracle has been generated as $\Pi_{a,b}^J$ and $\mathtt{ID}_a = \mathtt{ID}_j, \mathtt{ID}_b = \mathtt{ID}_i$ and $\Pi_{a,b}^J$ and $\Pi_{i,j}^t$ have the same session ID, then abort the game (**Event 3**).

  – If $SK_{i,j}^t \neq \perp$, return $SK_{i,j}^t$.

  – Otherwise,

* If $r_{i,j}^t \neq \perp$ (so $\ell_i \neq \perp$ and $D_i = \ell_i a P_1 = \ell_i \psi(aP_2)$), compute $K_{i,j}^t = \hat{e}(r_{i,j}^t Q_j, aP_2) \cdot \hat{e}(\ell_i \psi(aP_2), M)$ where Q_j is found from H_1^{list} for identifier $\mathtt{ID}_j$ and M is the received message on $tran_{i,j}^t$. By making an H_2 query, set $SK_{i,j}^t = H_2(\mathtt{ID}_i, \mathtt{ID}_j, r_{i,j}^t P_2, M, r_{i,j}^t M, K_{i,j}^t)$ if $\Pi_{i,j}^t$ is an initiator oracle, or $SK_{i,j}^t = H_2(\mathtt{ID}_j, \mathtt{ID}_i, M, r_{i,j}^t P_2, r_{i,j}^t M, K_{i,j}^t)$ otherwise, and update Ω by putting $SK_{i,j}^t$ then return $SK_{i,j}^t$ as the response.
* Otherwise, i.e. it should have $r_{i,j}^t = f_{i,j}^t a$ and $D_i = abP_1$. Algorithm $\mathcal{B}$ does not know both values and should not be able to compute $K_{i,j}^t = \hat{e}(f_{i,j}^t a Q_j, aP_2) \cdot \hat{e}(abP_1, M)$ and $f_{i,j}^t aM$ (note that the model requires that $i \neq j$). Algorithm $\mathcal{B}$ proceeds as follows:
  - Go through the list H_2^{list} to find a tuple $(\mathtt{ID}_i, \mathtt{ID}_j, f_{i,j}^t aP_2, M, Z_u, K_u, h_u)$ if $\mathtt{ID}_i$ is the initiator or a tuple $(\mathtt{ID}_j, \mathtt{ID}_i, M, f_{i,j}^t aP_2, Z_u, K_u, h_u)$ otherwise, meeting the equation $\hat{e}(\psi(f_{i,j}^t aP_2), M) = \hat{e}(P_1, Z_u)$ and for Type 2 or 4 pairings $\hat{e}(\psi(M), f_{i,j}^t aP_2) = \hat{e}(\psi(Z_u), P_2)$ as well.
  - If such Z_u is found, then compute

$$\begin{aligned} K_{i,j}^t &= \hat{e}(f_{i,j}^t a Q_j, R) \cdot \hat{e}(D_i, M) \\ &= \hat{e}(f_{i,j}^t a \ell_j P_1, aP_2) \cdot \hat{e}(abP_1, M), \text{ since } D_i = abP_1, Q_j = \ell_j P_1 \\ &= \hat{e}(f_{i,j}^t \ell_j \psi(aP_2), aP_2) \cdot \hat{e}(bP_1, \frac{1}{f_{i,j}^t} Z_u) \text{ since } f_{i,j}^t aM = Z_u. \end{aligned}$$

    and set $SK_{i,j}^t = H_2(\mathtt{ID}_i, \mathtt{ID}_j, f_{i,j}^t aP_2, M, Z_u, K_{i,j}^t)$ if oracle $\Pi_{i,j}^t$ is the initiator or $SK_{i,j}^t = H_2(\mathtt{ID}_j, \mathtt{ID}_i, M, f_{i,j}^t aP_2, Z_u, K_{i,j}^t)$ otherwise.
  - Otherwise, randomly sample $SK_{i,j}^t \in \{0,1\}^n$, put $(\mathtt{ID}_i, \mathtt{ID}_j, f_{i,j}^t aP_2, M, \Pi_{i,j}^t)$ if $\mathtt{ID}_i$ is the initiator or $(\mathtt{ID}_j, \mathtt{ID}_i, M, f_{i,j}^t aP_2, \Pi_{i,j}^t)$ into list $\mathcal{L}$.
  - $\mathcal{B}$ responds with $SK_{i,j}^t$ and updates Ω by putting $SK_{i,j}^t$.

- **Test**$(\Pi_{i,j}^t)$: If $t \neq J$ or ($t = J$ but) there is an oracle $\Pi_{j,i}^w$ with the same session ID with $\Pi_{i,j}^t$ that has been revealed, then $\mathcal{B}$ aborts the game (**Event 4**) (note that according to the rules of the game, $\Pi_{i,j}^t$ should have accepted, i.e. there are two messages on the oracle's transcript, so the check can be done properly). Otherwise ($\ell_j = \perp, Q_j = bP_1$ and $r_{i,j}^t = \perp$), $\mathcal{B}$ randomly chooses $\zeta \in \{0,1\}^n$ and responds to $\mathcal{A}$ with ζ.

Once $\mathcal{A}$ finishes queries and returns its guess, $\mathcal{B}$ proceeds with the following steps:

- Compute

$$D = \hat{e}(\ell_i \psi(aP_2), M),$$

  where M is the incoming message of $\Pi_{i,j}^J$ and note that because $i \neq j$ according to Definition 5.2.1, it has $D_i = \ell_i a P_1 = \ell_i \psi(aP_2)$ where $\ell_i \neq \perp$ is found from H_1^{list} corresponding to identifier $\mathtt{ID}_i$ of $\Pi_{i,j}^J$, and

$$\begin{aligned} K_{i,j}^t &= \hat{e}(cbP_1, aP_2) \cdot \hat{e}(\ell_i \psi(aP_2), M) \\ &= \hat{e}(P_1, P_2)^{abc} \cdot D \end{aligned}$$

- Algorithm $\mathcal{B}$ randomly chooses K_ℓ from H_2^{list} and returns K_ℓ/D as the response to the BDH challenge.

Claim 5.4.1. *If $\mathcal{B}$ did not abort the game, $\mathcal{A}$ could not find inconsistency between the simulation and the real world.*

Proof: The simulations of all the random oracles are valid and the messages of the oracles are uniformly distributed in the message space. Particularly, the simulator makes use of the programmability of random oracle H_2 and the pairing as the decisional algorithm of DH on $\mathbb{G}_2$ to keep the consistence of responses to the H_2 queries and the Reveal queries. So the adversary should not notice any difference from the real attack environment.

Claim 5.4.2. *Let **Event 5** be that $K = \hat{e}(cbP_1, aP_2) \cdot \hat{e}(\ell_i \psi(aP_2), M)$ was not queried on H_2. Then $\Pr[\overline{Event\ 5}] \geq \epsilon(k)$.*

Proof: Suppose in the game, the chosen fresh oracle for the Test query is $\Pi_{i,j}^t$ and party i is the initiator (the result holds if party i is the responder for the similar reason). Let $\mathcal{H}$ be the event that $(\mathtt{ID}_i, \mathtt{ID}_j, M_i^t, M_j^t, *, *)$ has been queried on H_2 where M_i^t is the message from party i and M_j^t is the message from party j or the adversary $\mathcal{A}$. Now let us consider the following situations:

- Situation 1. $\mathcal{A}$ reveals $\Pi_{u,v}^w$ where $u \notin \{i, j\}$ or $v \notin \{i, j\}$, i.e. IDs are different from the fresh oracle. Because $\mathtt{ID}_u$ and $\mathtt{ID}_v$, instead of $\mathtt{ID}_i$ and $\mathtt{ID}_j$, will be queried as the identifier to H_2, event $\mathcal{H}$ will not be caused by such Reveal queries. Recall that for the model defined in Section 5.2, it is required that $i \neq j$.

- Situation 2. $\mathcal{A}$ reveals $\Pi_{j,i}^w$ where j is the initiator (if i is the responder for the chosen fresh oracle, then j is the responder, i.e. party j has the same role in the w-th session as i in the t-th session). The Reveal query will only query in the form of $H_2(\mathtt{ID}_j, \mathtt{ID}_i, *, *, *, *)$, so event $\mathcal{H}$ will not be caused by this type of Reveal query.

- Situation 3. $\mathcal{A}$ reveals $\Pi_{j,i}^w$ where j is the responder (if i is the responder for the chosen fresh oracle, then j is the initiator, i.e. party i and j have matching roles in the sessions). In this case, H_2 could be queried on $(\mathtt{ID}_i, \mathtt{ID}_j, M_i^w, M_j^w, *, *)$. Because of the rule of the game, $\Pi_{j,i}^w$ has no matching conversation to $\Pi_{i,j}^t$, which means either $M_i^t \neq M_i^w$ or $M_j^t \neq M_j^w$ (recall that the session ID which is the message transcript of the session, is used to define matching conversations. Two oracles have matching conversation to each other only if both have the same message transcript). Hence event $\mathcal{H}$ will not be caused by this type of Reveal query.

- Situation 4. $\mathcal{A}$ reveals $\Pi_{i,j}^w$ for $w \neq t$ and party i plays different roles in the t-th and w-th session. As analysed in situation 2, event $\mathcal{H}$ will not be caused by this type of Reveal query.

- Situation 5. $\mathcal{A}$ reveals $\Pi_{i,j}^w$ for $w \neq t$ and party i plays the same role in the t-th and w-th session. Because for both oracles $\Pi_{i,j}^t$ and $\Pi_{i,j}^w$, i is not controlled by the

adversary, it is only negligibly likely that $M_i^t = M_i^w$ if the used random flips are generated uniformly. Hence event $\mathcal{H}$ will not (or with only negligible probability) be caused by this type of Reveal query.

Let $\mathcal{H}'$ be the event that $H_2(\mathtt{ID}_i, \mathtt{ID}_j, M_i^t, M_j^t, Z_{i,j}^t, K_{i,j}^t)$ was queried where $Z_{i,j}^t$ computed as the DH value and $K_{i,j}^t$ computed through pairings are the agreed secret of $\Pi_{i,j}^t$. We note here there are two possibilities that event $\mathcal{H}'$ happens in the proof of using random oracles.

- Case 1. The simulator, which could not compute $Z_{i,j}^t$ and $K_{i,j}^t$, was forced to respond with a random sample $SK_{i,j}^t$ in some Reveal query. Hence, although the oracle H_2 was not explicitly queried with the tuple value, in this case, we *regard* that $\mathcal{H}'$ has happened, as the random oracle should return a unique result corresponding to each input and if H_2 is later explicitly queried on $(\mathtt{ID}_i, \mathtt{ID}_j, M_i^t, M_j^t, Z_{i,j}^t, K_{i,j}^t)$, then $SK_{i,j}^t$ should be returned.

- Case 2. The adversary queried H_2 with $(\mathtt{ID}_i, \mathtt{ID}_j, M_i^t, M_j^t, Z_{i,j}^t, K_{i,j}^t)$.

As analysed above, we know that the Reveal queries won't cause $\mathcal{H}'$ with non-negligible probability. Hence Case 1 will only happen with negligible probability at most. In other words, event $\mathcal{H}'$ happens with non-negligible probability only if $\mathcal{A}$ has queried $H_2(\mathtt{ID}_i, \mathtt{ID}_j, M_i^t, M_j^t, Z_{i,j}^t, K_{i,j}^t)$.

As H_2 is a random oracle, we have $\Pr[\mathcal{A} \text{ wins}|\overline{\mathcal{H}'}] = \frac{1}{2}$. Then we have

$$\begin{aligned}
\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins}|\mathcal{H}']\Pr[\mathcal{H}'] + \Pr[\mathcal{A} \text{ wins}|\overline{\mathcal{H}'}]\Pr[\overline{\mathcal{H}'}] \\
&\leq \Pr[\mathcal{H}'] + \frac{1}{2}(1 - \Pr[\mathcal{H}']) = \frac{1}{2} + \frac{1}{2}\Pr[\mathcal{H}'].
\end{aligned}$$

$$\begin{aligned}
\Pr[\mathcal{A} \text{ wins}] &\geq \Pr[\mathcal{A} \text{ wins}|\overline{\mathcal{H}'}]\Pr[\overline{\mathcal{H}'}] \\
&= \frac{1}{2}(1 - \Pr[\mathcal{H}']) = \frac{1}{2} - \frac{1}{2}\Pr[\mathcal{H}'].
\end{aligned}$$

It follows that $\epsilon(k) = |2\Pr[\mathcal{A} \text{ wins}] - 1| \leq \Pr[\mathcal{H}']$, which means, $\mathcal{A}$ has computed $K_{j,i}^t$ with probability $\epsilon(k)$ and queried it on H_2.

Let **Event 6** be that, in the attack, adversary $\mathcal{A}$ indeed chose oracle $\Pi_{i,j}^J$ as the challenger oracle where $\mathtt{ID}_j$ was queried on H_1 as the I-th distinct identifier query. Then following the rules of the game defined in Section 5.2, it's clear that **Event 1, 2, 3, 4** would not happen and the game won't abort. So,

$$\Pr[\overline{(\texttt{Event 1} \vee \texttt{Event 2} \vee \texttt{Event 3} \vee \texttt{Event 4})}] = \Pr[\texttt{Event 6}] \geq \frac{1}{q_1 \cdot q_o}.$$

Let **Event 7** be that $\mathcal{B}$ found the correct K_ℓ. Overall, we have

$$\begin{aligned}
\Pr[\mathcal{B} \text{ wins}] &= \Pr[\texttt{Event 6} \wedge \overline{\texttt{Event 5}} \wedge \texttt{Event 7}] \\
&\geq \frac{1}{q_1 \cdot q_o \cdot q_2}\Pr[\overline{\texttt{Event 5}}] \\
&\geq \frac{1}{q_1 \cdot q_o \cdot q_2}\epsilon(k).
\end{aligned}$$

This concludes the proof. □

Theorem 5.4.2. *The SCK scheme has master key forward secrecy, provided the $DH^{\psi}_{2,2,2}$ assumption is sound and H_2 is modeled as random oracle. Specifically, suppose a PPT adversary $\mathcal{A}$ wins the game with non-negligible advantage $\epsilon(k)$. Then there exists a PPT algorithm $\mathcal{B}$ to solve the $DH^{\psi}_{2,2,2}$ problem with advantage*

$$Adv_{\mathcal{B}}^{DH^{\psi}_{2,2,2}} \geq \frac{1}{2}\epsilon(k).$$

Again the comment with respect to the computability of the function ψ holds.

Proof: Given a $\mathrm{DH}^{\psi}_{2,2,2}$ problem instance (aP_2, bP_2) on a set of pairing parameters, we construct an algorithm $\mathcal{B}$ to make use of $\mathcal{A}$ to compute abP_2. Algorithm $\mathcal{B}$ simulates the system setup to adversary $\mathcal{A}$ as follows: Randomly sample $s \in \mathbb{Z}_p$ and set the master public key to be $R = sP_2$ with the pairing parameters and two hash functions H_1 and H_2, and set the master secret key as s. The hash function H_2 will be modeled as a random oracle under the control of $\mathcal{B}$, and H_1 will be a normal cryptographic hash function. Moreover, the master secret key s is passed to $\mathcal{A}$ as well, so $\mathcal{B}$ no longer simulates the Corrupt query.

As in Theorem 5.4.1, we use $\Pi^t_{i,j}$ to represent the t-th one among all oracles created in the attack. Again algorithm $\mathcal{B}$ answers the following queries, which are asked by adversary $\mathcal{A}$ in an arbitrary order.

- $H_2(\mathtt{ID}^a_u, \mathtt{ID}^b_u, X_u, Y_u, Z_u, K_u)$: Algorithm $\mathcal{B}$ maintains an initially empty list H_2^{list} with entries of the form $(\mathtt{ID}^a_u, \mathtt{ID}^b_u, X_u, Y_u, Z_u, K_u, h_u)$. $\mathcal{B}$ responds to the query in the following way.
  - If a tuple indexed by $(\mathtt{ID}^a_u, \mathtt{ID}^b_u, X_u, Y_u, Z_u, K_u)$ is on the list, then $\mathcal{B}$ responds with h_u.
  - Otherwise, $\mathcal{B}$ goes through the list $\mathcal{L}$ (maintained in the Reveal query) with tuples of the form $(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, K^t_{i,j}, \Pi^t_{i,j})$ to find a tuple with values $(\mathtt{ID}^a_u, \mathtt{ID}^b_u, X_u, Y_u, K_u, \Pi^t_{i,j})$ and proceeds as follows (again, we assume X_u is the incoming message to $\Pi^t_{i,j}$):
    * Test if $\hat{e}(\psi(Y_u), X_u) = \hat{e}(P_1, Z_u)$ and for Type 2 or 4 pairing test as well $\hat{e}(\psi(X_u), Y_u) = \hat{e}(\psi(Z_u), P_2)$. If the equation holds then,
      - Find the value $SK^t_{i,j}$ from the list Ω.
      - Remove $(\mathtt{ID}^a_u, \mathtt{ID}^b_u, X_u, Y_u, K_u, \Pi^t_{i,j})$ from the list $\mathcal{L}$. Put $(\mathtt{ID}^a_u, \mathtt{ID}^b_u, X_u, Y_u, Z_u, K_u, SK^t_{i,j})$ in the list H_2^{list} and return $SK^t_{i,j}$. Note that $\Pi^t_{i,j}$ is put on the list $\mathcal{L}$ only when it has been revealed, so $SK^t_{i,j}$ has been sampled.
    * Otherwise (no tuple on $\mathcal{L}$ meets the test), algorithm $\mathcal{B}$ randomly chooses $h_u \in \{0,1\}^n$, inserts $(\mathtt{ID}^a_u, \mathtt{ID}^b_u, X_u, Y_u, Z_u, K_u, h_u)$ into the list and returns h_u.
  - Otherwise, $\mathcal{B}$ randomly chooses $h_u \in \{0,1\}^n$, inserts $(\mathtt{ID}^a_u, \mathtt{ID}^b_u, X_u, Y_u, Z_u, K_u, h_u)$ into the list and returns h_u.

- **Send($\Pi^t_{i,j}$, M)**: $\mathcal{B}$ maintains a list Ω for each oracle of the form ($\Pi^t_{i,j}, tran^t_{i,j}$, $f^t_{i,j}$, $K^t_{i,j}$, $SK^t_{i,j}$, $c^t_{i,j}$) where $tran^t_{i,j}$ is the transcript of the oracle so far; $f^t_{i,j}, c^t_{i,j}$ are used for special purpose explained below, and $K^t_{i,j}, SK^t_{i,j}$ are set $\perp$ initially. This list is updated in the Send query as well as in the Reveal query and H_2 query. $\mathcal{B}$ proceeds in the following way:
  - If M is not the second message on the transcript,
    * Randomly sample $f^t_{i,j} \in \mathbb{Z}_p$.
    * Randomly flip $c^t_{i,j} \in \{0,1\}$. If $c^t_{i,j} = 0$, set $V = f^t_{i,j} aP_2$, else $V = f^t_{i,j} bP_2$. If $V = P_2$, then responds to the DH challenge with $\frac{1}{f^t_{i,j}} bP_2$ if $c^t_{i,j} = 0$, or $\frac{1}{f^t_{i,j}} aP_2$ otherwise (**Event 1**).
    * If $M \neq \lambda$, compute
    $$K^t_{i,j} = \hat{e}(f^t_{i,j} H_1(\mathtt{ID}_j), saP_2) \cdot \hat{e}(sH_1(\mathtt{ID}_i), M),$$
    if $c^t_{i,j} = 0$, else set
    $$K^t_{i,j} = \hat{e}(f^t_{i,j} H_1(\mathtt{ID}_j), sbP_2) \cdot \hat{e}(sH_1(\mathtt{ID}_i), M)$$
    and accept the session.
    * Return V.
  - Otherwise, compute $K^t_{i,j} = \hat{e}(f^t_{i,j} H_1(\mathtt{ID}_j), saP_2) \cdot \hat{e}(sH_1(\mathtt{ID}_i), M)$ if $c^t_{i,j} = 0$, else compute $K^t_{i,j} = \hat{e}(f^t_{i,j} H_1(\mathtt{ID}_j), sbP_2) \cdot \hat{e}(sH_1(\mathtt{ID}_i), M)$, and accept the session.

- **Reveal($\Pi^t_{i,j}$)**: Algorithm $\mathcal{B}$ maintains a list $\mathcal{L}$ with tuples of the form ($\mathtt{ID}_i$, $\mathtt{ID}_j$, X_i, Y_j, $K^t_{i,j}$, $\Pi^t_{i,j}$). The algorithm $\mathcal{B}$ proceeds in the following way to respond:
  - Get the tuple of oracle $\Pi^t_{i,j}$ from Ω.
  - If $\Pi^t_{i,j}$ has not accepted, return $\perp$.
  - If the Test($\Pi^w_{a,b}$) query has been issued and if $\Pi^t_{i,j} = \Pi^w_{a,b}$, or $\mathtt{ID}_a = \mathtt{ID}_j$ and $\mathtt{ID}_b = \mathtt{ID}_j$ and two oracles have the same session ID, then disallow the query (this should not happen if the adversary obey the rules of the game).
  - If $SK^t_{i,j} \neq \perp$, return $SK^t_{i,j}$.
  - Otherwise,
    * Go through the list H_2^{list} to find a tuple ($\mathtt{ID}_i, \mathtt{ID}_j, M_i, M_j$, Z_u, $K^t_{i,j}$, h_u) if $\mathtt{ID}_i$ is the initiator or a tuple ($\mathtt{ID}_j, \mathtt{ID}_i$, M_j, M_i, Z_u, $K^t_{i,j}$, h_u) otherwise, meeting the equation $\hat{e}(\psi(M_i), M_j) = \hat{e}(P_1, Z_u)$ and for Type 2 or 4 pairing $\hat{e}(\psi(M_j), M_i) = \hat{e}(\psi(Z_u), P_2)$ as well, where M_i and M_j are the messages of party i and j in $tran^t_{i,j}$.
    * If such Z_u is found, then return $SK^t_{i,j} = h_u$.
    * Otherwise, randomly sample $SK^t_{i,j} \in \{0,1\}^n$, put ($\mathtt{ID}_i$, $\mathtt{ID}_j$, M_i, M_j, $K^t_{i,j}$, $\Pi^t_{i,j}$) if $\mathtt{ID}_i$ is the initiator or ($\mathtt{ID}_j$, $\mathtt{ID}_i$, M_j, M_i, $K^t_{i,j}$, $\Pi^t_{i,j}$) into list $\mathcal{L}$. $\mathcal{B}$ responds with $SK^t_{i,j}$ and puts $SK^t_{i,j}$ into Ω.

- **Test**$(\Pi^t_{i,j})$: By the rule of the game, there is a partner oracle $\Pi^w_{j,i}$ with the same session ID with $\Pi^t_{i,j}$ and both should not be revealed. $\mathcal{B}$ proceeds as follows:
  - Check if $c^t_{i,j} = c^w_{j,i}$. If it is true, then abort the game (**Event 2**).
  - Otherwise, without loosing generality, we assume $c^t_{i,j} = 0$ and $c^w_{j,i} = 1$, i.e. $M_i = f^t_{i,j} aP_2$ and $M_j = f^w_{j,i} bP_2$. $\mathcal{B}$ randomly chooses $\zeta \in \{0,1\}^n$ and responds to $\mathcal{A}$ with ζ.

Once $\mathcal{A}$ finishes queries and returns its guess, $\mathcal{B}$ proceeds with the following steps:

- For every pair (X_u, Y_u, Z_u) on H_2^{list} with $X_u = M_i, Y_u = M_j$, if the tested oracle $\Pi^t_{i,j}$ is an initiator oracle, otherwise with $X_u = M_j, Y_u = M_i$, check if $\hat{e}(\psi(X_u), Y_u) = \hat{e}(P_1, Z_u)$ and for Type 2 or 4 pairings, $\hat{e}(\psi(X_u), Y_u) = \hat{e}(\psi(Z_u), P_2)$ as well hold (M_i and M_j are found in $tran^t_{i,j}$). If no such Z_u meets the equation, abort the game (**Event 3**).

- Otherwise, return $\frac{1}{f^t_{i,j} \cdot f^w_{j,i}} Z_u$ as the response to the DH challenge.

Following similar arguments as in Theorem 5.4.1, we have following two claims:

Claim 5.4.3. *If $\mathcal{B}$ did not abort the game, $\mathcal{A}$ could not find inconsistency between the simulation and the real world.*

Claim 5.4.4. $\Pr[\overline{Event\ 3}] \geq \epsilon(k)$.

As $\Pr[\overline{\texttt{Event 2}}] = 1/2$, we have

$$\begin{aligned} \Pr[\mathcal{B} \text{ wins}] &= \Pr[\texttt{Event 1} \vee (\overline{\texttt{Event 2}} \wedge \overline{\texttt{Event 3}})] \\ &\geq \epsilon(k)/2. \end{aligned}$$

□

A few subtle points of the proofs are worth mentioning. First, the proofs need an extra operation, which is not explicitly specified in the protocol. Namely, the proofs assume that all values lie in the correct groups. This assumption has generally applied explicitly or implicitly in the literature. The importance of such check will be further discussed in Section 5.4.4. Second, we assume that xT_B (yT_A resp.) has a unique representation. This assumption has no significance in practice but is needed to keep the indistinguishability of the reductions from the real world. Third, for Type 4 pairings, there is negligible probability that the pairing is trivial. We ignored this issue in the reduction.

We note that by choosing $R = sP_1$ and the codomain of H_1 to be $\mathbb{G}_2$, there is another instantiation of the SCK scheme. We call it SCK′. SCK′ has better computation

performance and much shorter exchanged messages. On the other hand, it can only be implemented with Type 1, 3 and 4 pairings. The above proofs can be replicated for the SCK′ with Type 1 or 3 pairings based on assumption $\text{BDH}^{\psi}_{2,2,1}$ and $\text{DH}^{\psi}_{2,2,1}$ respectively with very little changes. The reduction for Type 4 pairing is more involved notationally, but the essential proof technique remains the same.

5.4.2 Security Analysis of the SYL Protocol

We now turn to considering security of the SYL scheme. Note that the SYL scheme can only be implemented when the isomorphism exists and hashing in $\mathbb{G}_2$ can be done efficiently. So the SYL scheme works only with Type 1 and Type 4 pairings. Here we formally present the proof for Type 1.

The security of the SYL scheme can be summarised by Theorem 5.4.3 and 5.4.4.

Theorem 5.4.3. *The SYL scheme is a secure AK, provided the BDH assumption is sound and the hash functions are modeled as random oracles. Specifically, suppose in the attack, a PPT adversary $\mathcal{A}$ which makes q_i queries to H_i for $i = 1, 2$ and creates q_o oracles, wins the game with non-negligible advantage $\epsilon(k)$. Then there exists a PPT algorithm $\mathcal{B}$ to solve the BDH problem with advantage*

$$Adv^{BDH}_{\mathcal{B}}(k) \geq \frac{1}{q_1 \cdot q_o \cdot q_2}\epsilon(k).$$

Proof: Given a BDH problem instance (P, aP, bP, cP), we construct an algorithm $\mathcal{B}$ using the adversary $\mathcal{A}$ against the protocol to solve the BDH problem.

$\mathcal{B}$ simulates the system setup to adversary $\mathcal{A}$ as follows. The system public parameters includes the pairing parameters of the input problem, $R = aP$ (hence $\mathcal{B}$ does not know the master secret key) and two functions H_1 and H_2 which are instantiated as random oracles under the control of $\mathcal{B}$.

Algorithm $\mathcal{B}$ randomly chooses $1 \leq I \leq q_1$ and $1 \leq J \leq q_o$ and starts simulating the real world where the adversary $\mathcal{A}$ launches the attack. Algorithm $\mathcal{B}$ answers the following queries, which are asked by adversary $\mathcal{A}$ in an arbitrary order.

- $H_1(\texttt{ID}_i)$: Algorithm $\mathcal{B}$ maintains an initially empty list H_1^{list} with entries of the form $(\texttt{ID}_i, Q_i, \ell_i)$. The algorithm $\mathcal{B}$ responds to the query in the following way.
  - If $\texttt{ID}_i$ already appears on H_1^{list} in a tuple $(\texttt{ID}_i, Q_i, \ell_i)$, then $\mathcal{B}$ responds with $H_1(\texttt{ID}_i) = Q_i$.

- Otherwise, if $\mathtt{ID}_i$ is the I-th unique identifier query, then $\mathcal{B}$ inserts $(\mathtt{ID}_i, bP, \perp)$ into the list and returns bP.
- Otherwise, $\mathcal{B}$ randomly chooses $\ell_i \in \mathbb{Z}_p$, inserts $(\mathtt{ID}_i, \ell_i P, \ell_i)$ into the list and returns $\ell_i P$.

- $H_2(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u)$: Algorithm $\mathcal{B}$ maintains an initially empty list H_2^{list} with entries of the form $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u, h_u)$. The algorithm $\mathcal{B}$ responds to the query in the following way.
  - If a tuple indexed by $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u)$ is on the list, then $\mathcal{B}$ responds with h_u.
  - Otherwise, $\mathcal{B}$ goes through the list $\mathcal{L}$ (maintained in the Reveal query) with tuples of the form $(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, \Pi_{i,j}^t)$ to find a tuple with values $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, \Pi_{i,j}^t)$ and proceeds as following:
    * Test if $\hat{e}(X_u, Y_u) = \hat{e}(P, Z_u)$. If the equation holds then,
      · Find the values $f_{i,j}^t$ and $SK_{i,j}^t$ corresponding to oracle $\Pi_{i,j}^t$ from the list Ω.
      · Find the value ℓ_j from H_1^{list} for party with $\mathtt{ID}_j$.
      · Compute the shared secret via the equation where $M = X_u$ if X_u is the incoming message to the oracle $\Pi_{i,j}^t$, or $M = Y_u$ otherwise,

$$\begin{aligned} K_{i,j}^t &= \hat{e}(M + Q_j, x_i R + D_i), \\ &= \hat{e}(M + \ell_j P, (f_{i,j}^t a)aP + abP), \text{ since } x_i = f_{i,j}^t a, D_i = abP \\ &= \hat{e}(M, f_{i,j}^t aaP) \cdot \hat{e}(M, abP) \cdot \hat{e}(\ell_j P, f_{i,j}^t aaP) \cdot \hat{e}(\ell_j P, abP), \\ &= \hat{e}(Z_u, aP) \cdot \hat{e}(\frac{1}{f_{i,j}^t} Z_u, bP) \cdot \hat{e}(f_{i,j}^t \ell_j aP, aP) \cdot \hat{e}(\ell_j bP, aP), \\ &\qquad \text{since } f_{i,j}^t aM = Z_u \\ &= \hat{e}(Z_u + f_{i,j}^t \ell_j aP + \ell_j bP, aP) \cdot \hat{e}(\frac{1}{f_{i,j}^t} Z_u, bP) \end{aligned}$$

    Note that $\Pi_{i,j}^t$ is put on the list $\mathcal{L}$ only when $\Pi_{i,j}^t$ has been revealed and $D_i = abP$, but $H_2(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_{i,j}^t)$ had not been queried before the Reveal query. So, $SK_{i,j}^t$ has been randomly sampled.
    · Set $h_u = SK_{i,j}^t$.
    · Remove $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, \Pi_{i,j}^t)$ from the list $\mathcal{L}$. Put $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_{i,j}^t, h_u)$ in the list H_2^{list}.
    · Check if $K_{i,j}^t = K_u$. If it is not true, $\mathcal{B}$ randomly chooses new $h_u \in \{0,1\}^n$, inserts $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u, h_u)$ into the list H_2^{list}.
    · Return h_u.

* Otherwise (no tuple on $\mathcal{L}$ meets the test), algorithm $\mathcal{B}$ randomly chooses $h_u \in \{0,1\}^n$, inserts $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u, h_u)$ into the list and returns h_u.

– Otherwise, $\mathcal{B}$ randomly chooses $h_u \in \{0,1\}^n$, inserts $(\mathtt{ID}_u^a, \mathtt{ID}_u^b, X_u, Y_u, Z_u, K_u, h_u)$ into the list and returns h_u.

- **Corrupt**($\mathtt{ID}_i$): $\mathcal{B}$ looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. $\mathcal{B}$ checks the value of ℓ_i: if $\ell_i \neq \perp$, then $\mathcal{B}$ responds with $\ell_i aP$; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).

- **Send**($\Pi_{i,j}^t$, **M**): $\mathcal{B}$ maintains a list Ω for each oracle of the form $(\Pi_{i,j}^t, tran_{i,j}^t, r_{i,j}^t, K_{i,j}^t, SK_{i,j}^t, f_{i,j}^t)$ where $tran_{i,j}^t$ is the transcript of the oracle so far; $r_{i,j}^t$ is the random integer used by the oracle to generate message, $f_{i,j}^t$ is used for special purpose explained below, and $K_{i,j}^t$ and $SK_{i,j}^t$ are set $\perp$ initially. Note that this list is updated in the Reveal query and H_2 query as well. $\mathcal{B}$ proceeds in the following way:

  – If M is the second message on the transcript, do nothing but simply accept the session. Otherwise,
  – Query $H_1(\mathtt{ID}_i)$ and $H_1(\mathtt{ID}_j)$.
  – If $t = J$,
    * If $\ell_j \neq \perp$, then abort the game (**Event 2**).
    * Otherwise, respond with cP and set $r_{i,j}^t = \perp$ (if $M = \lambda$, then party $\mathtt{ID}_i$ is an initiator, otherwise a responder as M is the first message of the session).
  – Otherwise,
    * If $\ell_i = \perp$, randomly choose $f_{i,j}^t \in \mathbb{Z}_p$ and respond with $f_{i,j}^t aP$ and set $r_{i,j}^t = \perp$.
    * Otherwise, randomly choose $r_{i,j}^t \in \mathbb{Z}_p$ and respond with $r_{i,j}^t P$.

- **Reveal**($\Pi_{i,j}^t$): $\mathcal{B}$ maintains a list $\mathcal{L}$ with tuples of the form $(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, \Pi_{i,j}^t)$, $\mathcal{B}$ proceeds in the following way to respond:

  – Get the tuple corresponding to oracle $\Pi_{i,j}^t$ from Ω.
  – If oracle $\Pi_{i,j}^t$ has not accepted, then respond with $\perp$.
  – If $t = J$ or if the J-th oracle has been generated as $\Pi_{a,b}^J$ and $\mathtt{ID}_a = \mathtt{ID}_j, \mathtt{ID}_b = \mathtt{ID}_i$ and $\Pi_{a,b}^J$ and $\Pi_{i,j}^t$ have the same session ID, then abort the game (**Event 3**).
  – If $SK_{i,j}^t \neq \perp$, return $SK_{i,j}^t$.
    * If $r_{i,j}^t \neq \perp$ (so $\ell_i \neq \perp$ and $D_i = \ell_i aP$),
      · Compute $K_{i,j}^t = \hat{e}(M + Q_j, (r_{i,j}^t + \ell_i)aP)$ where Q_j is found from H_1^{list} for identifier $\mathtt{ID}_j$ and M is the received message on $tran_{i,j}^t$. Set $SK_{i,j}^t = H_2(\mathtt{ID}_i, \mathtt{ID}_j, r_{i,j}^t P, M, r_{i,j}^t M, K_{i,j}^t)$ if $\Pi_{i,j}^t$ is an initiator oracle, or $SK_{i,j}^t = H_2(\mathtt{ID}_j, \mathtt{ID}_i, M, r_{i,j}^t P, r_{i,j}^t M, K_{i,j}^t)$ otherwise, and return $SK_{i,j}^t$ as the response.

* Otherwise, i.e. it should have $r_{i,j}^t = f_{i,j}^t a$ and $D_i = abP$. Algorithm $\mathcal{B}$ does not know both values and should compute $K_{i,j}^t = \hat{e}(M + \ell_j P, f_{i,j}^t aR + D_i)$ and $f_{i,j}^t aM$ (note that the model requires that $i \neq j$). Algorithm $\mathcal{B}$ proceeds as follows:
  - Go through the list H_2^{list} to find a tuple $(\mathtt{ID}_i, \mathtt{ID}_j, f_{i,j}^t aP, M, Z_u, K_u, h_u)$ if $\mathtt{ID}_i$ is the initiator or a tuple $(\mathtt{ID}_j, \mathtt{ID}_i, M, f_{i,j}^t aP, Z_u, K_u, h_u)$ otherwise, meeting the equation $\hat{e}(M, f_{i,j}^t aP) = \hat{e}(P, Z_u)$.
  - If such Z_u is found, then compute

$$\begin{aligned} K_{i,j}^t &= \hat{e}(M + \ell_j P, f_{i,j}^t aR + D_i) \\ &= \hat{e}(\frac{1}{f_{i,j}^t a} Z_u + \ell_j P, f_{i,j}^t aaP + abP) \text{ since } M = \frac{1}{f_{i,j}^t a} Z_u, \\ &= \hat{e}(Z_u, aP) \cdot \hat{e}(\frac{1}{f_{i,j}^t} Z_u, bP) \cdot \hat{e}(f_{i,j}^t \ell_j aP, aP) \cdot \hat{e}(\ell_j bP, aP), \\ &= \hat{e}(Z_u + f_{i,j}^t \ell_j aP + \ell_j bP, aP) \cdot \hat{e}(\frac{1}{f_{i,j}^t} Z_u, bP) \end{aligned}$$

    and return $SK_{i,j}^t = H_2(\mathtt{ID}_i, \mathtt{ID}_j, f_{i,j}^t aP, M, Z_u, K_{i,j}^t)$ if party i is the initiator or $SK_{i,j}^t = H_2(\mathtt{ID}_j, \mathtt{ID}_i, M, f_{i,j}^t aP, Z_u, K_{i,j}^t)$ if i is the responder.
  - Otherwise, randomly sample $SK_{i,j}^t \in \{0,1\}^n$, put $(\mathtt{ID}_i, \mathtt{ID}_j, f_{i,j}^t aP, M, \Pi_{i,j}^t)$ if $\mathtt{ID}_i$ is the initiator or $(\mathtt{ID}_j, \mathtt{ID}_i, M, f_{i,j}^t aP, \Pi_{i,j}^t)$ into list $\mathcal{L}$. $\mathcal{B}$ responds with $SK_{i,j}^t$.

- **Test**$(\Pi_{i,j}^t)$: If $t \neq J$ or $(t = J$ but) there is an oracle $\Pi_{j,i}^w$ with the same session ID with $\Pi_{i,j}^t$ that has been revealed, then $\mathcal{B}$ aborts the game (**Event 4**). Otherwise $(\ell_j = \perp, Q_j = bP$ and $r_{i,j}^t = \perp)$, $\mathcal{B}$ randomly chooses $\zeta \in \{0,1\}^n$ and responds to $\mathcal{A}$ with ζ.

Once $\mathcal{A}$ finishes queries and returns its guess, $\mathcal{B}$ proceeds with the following steps:

- For every pair (X_u, Y_u, Z_u) on H_2^{list} with $X_u = cP, Y_u = M$ if the tested oracle $\Pi_{i,j}^t$ is an initiator oracle, otherwise with $X_u = M, Y_u = cP$ where M is the received message on $tran_{i,j}^t$, check if $\hat{e}(X_u, Y_u) = \hat{e}(P, Z_u)$ holds. If no such Z_u meets the equation, abort the game (**Event 5**).

- Otherwise, compute

$$D = \hat{e}(\ell_i(bP + M) + Z_u, aP).$$

  Note that because $i \neq j$ according to Definition 5.2.1, it has $D_i = \ell_i aP$ where $\ell_i \neq \perp$

found from H_1^{list} corresponding to identifier $\mathtt{ID}_i$ and

$$\begin{aligned} K_{i,j}^t &= \hat{e}(M + Q_j, x_i R + D_i), \\ &= \hat{e}(M + bP, caP + \ell_i aP) \\ &= \hat{e}(P,P)^{abc} \cdot \hat{e}(bP, \ell_i aP) \cdot \hat{e}(M, caP + \ell_i aP), \\ &= \hat{e}(P,P)^{abc} \cdot \hat{e}(\ell_i bP, aP) \cdot \hat{e}(Z_u, aP) \cdot \hat{e}(\ell_i M, aP) \text{ since } M = \frac{1}{c} Z_u, \\ &= \hat{e}(P,P)^{abc} \cdot D. \end{aligned}$$

- Algorithm $\mathcal{B}$ randomly chooses K_ℓ from H_2^{list} and returns K_ℓ / D as the response to the BDH challenge.

Following the similar argument as in Theorem 5.4.1, we have following two claims.

Claim 5.4.5. *If $\mathcal{B}$ did not abort the game, $\mathcal{A}$ could not find inconsistency between the simulation and the real world.*

Claim 5.4.6. *Let **Event 6** be that $K = \hat{e}(M + bP, (c + \ell_i)aP)$ was not queried on H_2. Then* $\Pr[\overline{Event\ 5} \wedge \overline{Event\ 6}] \geq \epsilon(k)$.

Let **Event 7** be that, in the attack, adversary $\mathcal{A}$ indeed chose oracle $\Pi_{i,j}^J$ as the challenger oracle where $\mathtt{ID}_j$ was queried on H_1 as the I-th distinct identifier query. Then following the rules of the game defined in Section 5.2, it's clear that **Event 1, 2, 3, 4** would not happen. So,

$$\Pr[\overline{(\mathtt{Event\ 1} \vee \mathtt{Event\ 2} \vee \mathtt{Event\ 3} \vee \mathtt{Event\ 4})}] = \Pr[\mathtt{Event\ 7}] \geq \frac{1}{q_1 \cdot q_o}.$$

Let **Event 8** be that $\mathcal{B}$ did not abort in the game. Let **Event 9** be that $\mathcal{B}$ found the correct K_ℓ. Overall, we have

$$\begin{aligned} \Pr[\mathcal{B}\ \text{wins}] &= \Pr[\mathtt{Event\ 8} \wedge \overline{\mathtt{Event\ 6}} \wedge \mathtt{Event\ 9}] \\ &= \Pr[\mathtt{Event\ 7} \wedge \overline{\mathtt{Event\ 5}} \wedge \overline{\mathtt{Event\ 6}} \wedge \mathtt{Event\ 9}] \\ &\geq \frac{1}{q_1 \cdot q_o \cdot q_2} \Pr[\overline{\mathtt{Event\ 5}} \wedge \overline{\mathtt{Event\ 6}}] \\ &\geq \frac{1}{q_1 \cdot q_o \cdot q_2} \epsilon(k). \end{aligned}$$

This concludes the proof. □

Theorem 5.4.4. *The SYL scheme has master key forward secrecy, provided the DH assumption on $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$ holds and H_2 is modeled as random oracle. Specifically, suppose a PPT adversary $\mathcal{A}$ wins the game with non-negligible advantage $\epsilon(k)$. Then there exists a PPT algorithm $\mathcal{B}$ to solve the DH problem in $\mathbb{G}$ with advantage*

$$Adv_{\mathcal{B}}^{DH} \geq \frac{1}{2} \epsilon(k).$$

Proof: The proof is very similar to Theorem 5.4.2. Here we only specify the difference from the proof in Theorem 5.4.2 which is mainly how the agreed secret K is computed.

- **Send($\Pi^t_{i,j}$, M)**:
  - If M is not the second message on the transcript,
    - If $M \neq \lambda$, compute $K^t_{i,j} = \hat{e}(M + H_1(\mathtt{ID}_j), V + H_1(\mathtt{ID}_i))^s$ and accept the session.
  - Otherwise, compute $K^t_{i,j} = \hat{e}(M + H_1(\mathtt{ID}_j), f^t_{i,j}aP + H_1(\mathtt{ID}_i))^s$ if $c^t_{i,j} = 0$, else compute $K^t_{i,j} = \hat{e}(M + H_1(\mathtt{ID}_j), f^t_{i,j}bP + H_1(\mathtt{ID}_i))^s$, and accept the session. □

Similar to Theorem 5.4.2, one can aslo establish a reduction by allowing the adversary to setup the system parameters.

Following the observation of the security model in Section 5.2, we can confirm that the SCK scheme and the SYL scheme achieve very strong security properties including implicit mutual key authentication, known session key security, key compromise impersonation resilience, unknown key share resilience and master key forward secrecy.

5.4.3 The Built-in Decisional Function

To the best of our knowledge, so far the two proofs above are the only two security reductions which are constructed successfully based on the weakest complexity assumption (the BDH assumption) possible for this type of protocol using the Diffie-Helmman exchange with the security property of key-compromise impersonation resilience.

The first formal security analysis of an IB-KAP from pairings in the random oracle model was given by Chen and Kudla [63]. In the first version of their proof, Chen and Kudla claimed that the CK scheme is secure in the Bellare-Rogaway model. Later on we pointed out a flaw in their proof in dealing with reveal queries. They then corrected this error by modifying the proof under a weaker variant of the Bellare-Rogaway model, where the adversary is not allowed to make *any* reveal queries [62]. After that, a number of other protocols in this type were analysed in this weak model as well, for example, the MB scheme [124]. Note that the reduction of the MB scheme in [124, 125] is invalid even in the weak model and the detailed discussion on this issue is given in Section 5.5. Choo, Boyd.

and Hitchcock in [43] revisited the CK scheme and MB scheme and demonstrated that with slight change these two protocols can be proven secure in the random oracle model with a looser restriction that the adversary is not allowed to make reveal queries to a number of selected sessions, but allowed to other sessions. Their contribution improved the CK and MB scheme and their security analysis, and can also benefit other schemes. However, since a reveal query captures the known session key security property, neither the full nor partial restriction of disallowing reveal queries is really acceptable. The BMP scheme in [26] can be proven based on the BDH assumption to hold a number of security properties but without the key-compromise impersonation resilience.

In the security proof of this type of protocol following the model defined in Section 5.2, a simulator S of a real protocol has a goal to solve a pairing related hard problem, while an adversary $\mathcal{A}$ has a goal to break the protocol. The reason that S has the difficulty to answer reveal queries to certain sessions is that, without solving a hard computational problem, the simulator normally cannot compute the session secrets in these sessions. This property is required on purpose in this type of protocols, otherwise these protocols may not have the security property of key-compromise impersonation resilience. In the follow we explain this in details.

In general, three types of secrets are used in this type of key agreement protocols: the KGC master secret key, each party's identity-based private key, which is computed using the master secret key and the party's identifier, and each party's ephemeral secret, which is used to compute the party key token. In a protocol, after exchanging the key token with the other party, each party takes as input the master public key, its own identity-based private key and ephemeral secret along with the other party's identifier and the key token, and computes a pairing (or a few pairings) as a session secret shared with the other party.

To build a proof based on the BDH assumption, S's goal is to solve a random BDH problem, i.e. given $(xP,\ yP,\ zP)$ for $x, y, z \in \mathbb{Z}_p$, S has to compute $\hat{e}(P,P)^{xyz}$. In the

game, algorithm S arranges these three secrets as follows: The master secret key is x; the identity-based public key of the attacked party (say I) is related to y (so S does not know I's private key); the ephemeral secret of the challenge party (say J) is z. To answer reveal queries from $\mathcal{A}$, S has to deal with the following different sessions:

- Challenge session$(\Pi^s_{I,J}, \Pi^t_{J,I})$: $\mathcal{A}$ impersonates oracle $\Pi^s_{I,J}$ and challenges oracle $\Pi^t_{J,I}$. S does not need to answer any reveal query to this session, based on the definition of the security model in Section 5.2.

- Session $(\Pi^s_{I,J}, \Pi^t_{J,I})$: $\mathcal{A}$ impersonates oracle $\Pi^t_{J,I}$ and asks a reveal query to oracle $\Pi^s_{I,J}$. S is not able to compute the session secret because it does not know I's private key; otherwise, the session secret can be computed by using the party ephemeral secret and the partner long-term key, and therefore the key-compromise impersonation resilience property does not hold in this protocol.

- Session $(\Pi^s_{I,C}, \Pi^t_{C,I})$: $\mathcal{A}$ impersonates oracle $\Pi^t_{C,I}$ $(C \notin \{I, J\})$ and asks a reveal query to oracle $\Pi^s_{I,C}$. Again, S is not able to compute the session secret for the same reason as the above session.

- Session $(\Pi^s_{I,C}, \Pi^t_{C,I})$: $\mathcal{A}$ impersonates oracle $\Pi^s_{I,C}$ and asks a reveal query to oracle $\Pi^t_{C,I}$. S can compute the session secret by following the protocol correctly.

- Session $(\Pi^s_{C,D}, \Pi^t_{D,C})$: $\mathcal{A}$ impersonates either oracle $\Pi^s_{C,D}$ or $\Pi^t_{D,C}$ and asks a reveal query to the other oracle, where C and D is not I. S can compute the session secret by following the protocol correctly.

Computing the session secret guarantees that S can answer the reveal queries, but is not necessary when the security proof is composed in the random oracle model. In the random oracle model, when it controls a random oracle, which takes as input the session secret and outputs a random number as the session key, S can choose a random number as the answer

to a reveal query even though it cannot compute the session secret. The problem is that for some sessions, as Sessions $(\Pi^s_{I,J}, \Pi^t_{J,I})$ and $(\Pi^s_{I,C}, \Pi^t_{C,I})$ discussed above, S cannot compute a session secret, but $\mathcal{A}$ may be able to do so. Therefore, in order to check whether or not S acts as a real protocol, $\mathcal{A}$ can query the random oracle with the correct session secret after the reveal query. If S cannot recognize this correct value, it cannot make the output of the random oracle consistent with the responses to the reveal query.

This tells us that the computation of a session secret in the reduction can be replaced with a related decisional operation in the random oracle model. As long as S is able to make a right decision to $\mathcal{A}$'s random oracle query, the behavior of S, from $\mathcal{A}$'s point of view, is indistinguishable from the real world. Researchers have tried a number of various ways to solve this problem. Let us take a closer look at them.

In the early attempt of solving the problem, we used a coin query to force the adversary to register the ephemeral secret used to generate the key token with the simulator [64]. The model with the coin query can address certain attacks which are not covered in the model with the reveal query completely disallowed. On the other hand, for some attacks, the adversary may not know the ephemeral value corresponding to the key token used in the attacks. Hence, the model requires a protocol to be analysed in two separate reductions, one with the reveal query disallowed and the other with both the coin query and the reveal query allowed. However, this approach still does not guarantee the security of a protocol even if both valid reductions in the model can be constructed.

Kudla and Paterson in [112] proposed a modular proof approach, which introduces a decisional oracle in the security proof. By resorting to this decisional oracle, the simulator can choose a random number to answer a reveal query and maintain all random answers consistent to each other. This approach has been used to prove secure a number of protocols such as the Smart protocol [109] under a gap assumption. However, this approach has to rely on the assumption that such a decisional oracle exists and the gap problem is sound.

The former may not be true in this type of protocols, and the latter may not be as strong as the computational problem.

Wang in [161] proposed an approach, which is opposite to the Kudla and Paterson one. This approach has to resort to a computational oracle and is used to analyse the Wang scheme [161] based on a decisional assumption instead. As in the Kudla and Paterson approach, the problem of relying on an oracle, which nobody knows how to construct using any polynomial algorithm in the real world, also happens in this approach. In addition, using this approach the simulator has to guarantee that the computational oracle would not be bullied by the adversary to compute the underlying hard problem challenge.

It is worth mentioning a new proof technique introduced recently by Gentry [82]. The technique enables us to provide proofs for certain IBEs without random oracles. Generally in a game of IBE, a simulator like the simulator in the key agreement game, is faced with the similar difficulty that it has to answer certain decryption query but without knowing the corresponding private key. However, by applying the Gentry's technique the simulator can compute a valid private key of every party involved in the game. Hence all the decryption queries can be answered correctly. The same technique can be applied to key agreements with similar key construction. However the technique substantially relies on two bases: (1) a stronger intractability assumption like ℓ-BDHI and (2) a nondeterministic Extract algorithm different from ones in the SOK and SK key constructions.

If the underlying assumption fails, the proof becomes meaningless. This means that a proof of security is worth more when the assumption is weaker. To improve the above solutions for a complete proof with the weakest possible assumption, we have adopted a new approach in the above proofs, which incorporates a built-in decisional function. With this function, the simulator can now take the advantage of the "help" of the adversary either to compute the session secret or to maintain the consistency of random oracle answers. Such a built-in decisional function can be constructed by including a DH key token computed

with the ephemeral secret in the exchanged message and including the DH key value as part of the established session secret. Based on the fact that the DDH problem is not hard in the pairing-friendly groups, the simulator in the new approach does not need to rely on an outside computational oracle (when the isomorphism ψ is efficiently computable) in order to generate the session key to be revealed as required in the Wang approach, or an outside decisional oracle to keep the consistency between the random oracle queries and the reveal queries as required in the Kudla and Paterson method, or the knowledge of the adversary ephemeral secret as required in the approach of [64]. As a result of incorporating the built-in decisional function, the security reduction can be constructed on the weakest possible assumptions for this type of protocols. This is not just playing a proof trick in the random oracle. The approach indeed guides the protocol design. By including the DH key value as part of the session secret, the adversary is forced to know at least one ephemeral secret of a DH token to compute the session key. This significantly constrains the adversary's attacking methods.

To be specific, in the above two protocols, to introduce an efficient built-in decisional function into the protocols, the DH key value xyP_2 and the pairing-computed key K are used together in the session key computation through a random oracle H_2. The decisional function then is constructed as $\hat{e}(\psi(X), Y) \stackrel{?}{=} \hat{e}(P_1, Z)$ where X, Y are the exchanged key tokens and Z is the DH key value input of H_2. Now the simulator surely can make use of the decisional function to find out that the session secret including xyP_2 and K has not been queried, if xyP_2 has not been queried on H_2. When the DH value has indeed been queried on H_2, the simulator has to compute K, otherwise, it will have to resort to a BDH decisional oracle as required by the Kudla-Paterson method. By noticing that the difficulty of computing K in some sessions is to compute $\hat{e}(\psi(abP_2), Y)$ for some $Y = yP_2$ generated by the adversary, the simulator makes use of its freedom of choosing some ephemeral secret x for $X = xP_2$ to set $X = raP_2$ (or $X = rbP_2$) in those sessions. Now with the value

$Z = raY$ (or $Z = rbY$) found by the decisional function and value r, K can be computed as $\hat{e}(\psi(abP_2), Y) = \hat{e}(\psi(bP_2), \frac{1}{r}Z) = \hat{e}(bP_1, \frac{1}{r}Z)$ (or $\hat{e}(\psi(aP_2), \frac{1}{r}Z)$). So, with the decisional function the simulator now can either find out the session secret has not been queried with the random oracle or compute the agreed secret value from the correct DH key value.

The same approach can be applied to prove the security of some other schemes.

5.4.4 Group Membership Testing

In almost all key agreement schemes with Diffie-Hellman tokens, an assumption is made that tokens passed from one party to another lie in the correct groups. Such assumptions are often implicit within security proofs. However, one either needs to actually check that given tokens lie within the correct group, or force the tokens to lie in the group via additional computation, or by choosing parameters carefully so that the problem does not arise. Indeed some attacks on key agreement schemes either in practice or in principle are possible because implementors do not test for group membership, for example the small subgroup attack [115] and the even more sophisticated attacks [123, 129].

The proofs in this section and following sections also ask for checking the group membership of exchanged tokens. To demonstrate the importance of such check, we present an attack on the optimised SYL protocol. Suppose one optimises the SYL protocol with Type 4 pairings in the following way.

$$A \rightarrow B : T_A = xP_2$$
$$B \rightarrow A : T_B = yP_1$$

Upon completion of the message exchange, A computes $xT_B = xyP_1$ and $K = \hat{e}(T_B + \psi(Q_B), xR + D_A)$ and B computes $y\psi(T_A) = xyP_1$ and $K = \hat{e}(y\psi(R) + \psi(D_B), T_A + Q_A)$. Recall that $R = sP_2 = s\frac{1}{k}\mathcal{P}_1 + s\mathcal{P}_2$ is the master public key and s is the master secret key and k is the embedding degree here. This optimised version has smaller bandwidth cost and better performance than the original SYL specification. Suppose to save time that party B

does not check whether T_A lies in the subgroup generated by P_2, but only whether it lies in $\mathbb{G}_2$, i.e. it computes some of the subgroup membership test but not all of it. We show in this situation that an adversary C can impersonate party A to B.

Without losing generality, we assume

$$Q_A = a\mathcal{P}_1 + b\mathcal{P}_2 \in \mathbb{G}_2 \text{ and } Q_B = c\mathcal{P}_1 + d\mathcal{P}_2 \in \mathbb{G}_2.$$

Adversary C chooses random integers x and z from $\mathbb{Z}_p^*$, and generates the message T_A in the following attack.

$$\begin{aligned} C_A \rightarrow B &: T_A = x\tfrac{1}{k}\mathcal{P}_1 - b\mathcal{P}_2 + z\mathcal{P}_2 \\ B \rightarrow C_A &: T_B = yP_1 = y\mathcal{P}_1 \end{aligned}$$

Now B computes $y\psi(T_A) = yxP_1 = xT_B$ and

$$\begin{aligned} K &= \hat{e}(y\psi(R) + \psi(d_B), T_A + Q_A) \\ &= \hat{e}(ys\mathcal{P}_1 + \psi(sQ_B), x\frac{1}{k}\mathcal{P}_1 - b\mathcal{P}_2 + z\mathcal{P}_2 + a\mathcal{P}_1 + b\mathcal{P}_2) \\ &= \hat{e}(ys\mathcal{P}_1 + s\psi(Q_B), z\mathcal{P}_2) \\ &= \hat{e}(T_B + \psi(Q_B), s\mathcal{P}_2)^z \\ &= \hat{e}(T_B + \psi(Q_B), R)^z \end{aligned}$$

Both xT_B and $K = \hat{e}(T_B + \psi(Q_B), R)^z$ can be computed by C.

To prevent this attack, party B should also check that for $T_A = x_1\frac{1}{k}\mathcal{P}_1 + x_2\mathcal{P}_2$, $x_1 = x_2$, i.e. T_A is in the cyclic group generated by P_2. A careful analysis of the proof of the SYL protocol reveals that for the above optimization one is unable to obtain a proof if full group membership testing is not performed.

Group membership testing in $\mathbb{G}_1 = \mathcal{G}_1$, $\mathbb{G}_t$, $\mathcal{G}$, and $\mathcal{G}_2$ can be done in the standard way via multiplication and by inspection of the representation. If the cofactor is smaller than p, one can test membership via cofactor multiplication. However, in many pairing-based situations the cofactor is larger than p, in which case membership of the group of

exponent p is tested by multiplication by p. Note that depending on the security parameter, this membership test may be quite expensive, as one may need to perform quite a large multiplication.

In the Type 2 and 4 situations, there are other group tests that may need to be performed, which cannot be performed as above, namely testing whether a given point Q is a multiple of $P_2 = \frac{1}{k}\mathcal{P}_1 + \mathcal{P}_2$. In other words we wish to test whether $Q \in \langle P_2 \rangle$. We first test whether Q has order p by testing, via multiplication as above, whether it is in $\mathcal{G}$. Then we write $Q = a\mathcal{P}_1 + b\mathcal{P}_2$, for unknown a and b; one can compute $a\mathcal{P}_1$ and $b\mathcal{P}_2$ from Q via

$$a\mathcal{P}_1 = \frac{1}{k}\mathrm{Tr}(Q) \text{ and } b\mathcal{P}_2 = Q - a\mathcal{P}_1,$$

which requires one multiplication in $\mathbb{G}_1$. We need to test whether $a = b/k$, which can be done by performing the following test

$$\begin{aligned} \hat{e}(\mathrm{Tr}(Q), \mathcal{P}_2) &= \hat{e}(ka\mathcal{P}_1, \mathcal{P}_2) = \hat{e}(\mathcal{P}_1, b\mathcal{P}_2) \\ &= \hat{e}(\mathcal{P}_1, Q - \frac{1}{k}\mathrm{Tr}(Q)). \end{aligned}$$

In the Type 4 situation, another situation occurs when we wish to test whether a point $Q = a\mathcal{P}_1 + b\mathcal{P}_2$ is a multiple of a point $P = c\mathcal{P}_1 + d\mathcal{P}_2$ without knowing a, b, c or d. We first test whether $P, Q \in \mathcal{G}$ as above. Then we test whether $a = tc$ and $b = td$ for some unknown t by testing whether

$$\hat{e}(\mathrm{Tr}(Q), P - \frac{1}{k}\mathrm{Tr}(P)) = \hat{e}(\mathrm{Tr}(P), Q - \frac{1}{k}\mathrm{Tr}(Q)).$$

5.5 Security Analysis of the McCallugh-Barreto Protocol

Using the SK key construction, McCullagh and Barreto presented an identity-based KAP at CT-RSA 2005 [125], which appears to be the most efficient one in computation among the existing proposals of this type. However the scheme is vulnerable to a key-compromise impersonation attack. In order to recover from this security weakness, McCullagh and

Barreto [124], and Xie [163] independently proposed two fixes. Attempting to demonstrate the security of the schemes, they provided a security reduction for each protocol in the weak variant of the Bellare-Rogaway's key agreement model without the Reveal query as we noted already in Section 5.4.3.

In this section, we revisit the security proofs in [125, 124, 163] and show that all these three proofs are problematic. More specifically, in their security reductions, the property of indistinguishability between their simulation and the real world was not held. We note that in any identity-based cryptographic scheme, given a certain identity string and the system parameters, it is universally verifiable whether the private key corresponding to the identity string is correctly constructed or not. Therefore, if a simulator of the real world where an adversary launches attack, is not able to offer the adversary necessary evidence, which allows the adversary to verify the correction of a simulated key construction, the simulation fails, because the adversary can immediately notice the inconsistency between the simulation and the simulated real world. All the three proofs failed to provide this feature.

As pointed out in [119], Xie's scheme still suffers from the key-compromise impersonation attack and further a trivial man-in-the-middle attack, hence here we do not formally analyse it. Instead, we slightly modify McCullagh and Barreto's second protocol [124] and then present a formal reduction for the scheme in the key agreement model presented in Section 5.5.2. The new reduction demonstrates the security strength of the scheme.

5.5.1 The MB Protocol and its Variants

Here we recall McCullagh and Barreto's protocol and its variants. These protocols use the same key construction (the SK key construction) and exchange the same message flows. However, in the last step of the protocols, each protocol has a different scheme to compute an established session key.

Setup. The KGC executes the following operations:

1. On input 1^k, generate a set of pairing parameters of the required size.
2. Pick a random $s \in \mathbb{Z}_p^*$ and compute $R = sP_1$.
3. Pick two cryptographic hash functions as follows:

$$H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*,$$
$$H_2 : \mathbb{G}_t \rightarrow \{0,1\}^n$$

   for some integer n.
4. Set s as the master secret key which is kept secret by KGC and publish others as the system public parameters (also called the master public key).

Extract. The schemes employ the SK key construction [151]. Given an identity $\mathtt{ID}_A$, the master secret key s, and the system public parameters, the algorithm computes $H_1(\mathtt{ID}_A) = \alpha \in \mathbb{Z}_p^*$ and the corresponding private key $D_A = \frac{1}{s+\alpha} P_2$ for $\mathtt{ID}_A$. $Q_A = \alpha P_1 + sP_1$ will be treated as the real public key corresponding to $\mathtt{ID}_A$.

Protocol. Suppose $H_1(A) = \alpha$ and $H_1(B) = \beta$. Party A and B randomly choose x and y from $\mathbb{Z}_p^*$ respectively. The protocol proceeds as follows.

$$A \rightarrow B : T_A = xQ_B = x(\beta P_1 + sP_1)$$
$$B \rightarrow A : T_B = yQ_A = y(\alpha P_1 + sP_1)$$

On completion of the protocol, there are three ways to compute the agreed secret, and each has different security strength. Here we slightly change the protocols in [125, 124, 163] by employing an extra hash function H_2 on the agreed secret to generate the session keys. In the original schemes described in [125, 124, 163], the use of H_2 is not explicitly required. It will result in a potential security problem, which will be discussed in Section 5.5.2.

MB-1 (McCullagh-Barreto's first scheme (MB-1) [125]). A computes $K = \hat{e}(T_B, D_A)^x = \hat{e}(P_1, P_2)^{xy}$ and B computes $K = \hat{e}(T_A, D_B)^y = \hat{e}(P_1, P_2)^{xy}$. The agreed session key is

$SK = H_2(\hat{e}(P_1, P_2)^{xy})$. This scheme *appears* to provide an interesting security property: the perfect forward secrecy. However, this scheme does not achieve the key-compromise impersonation resilience.

A KCI attack works as in Figure 5.2 where adversary E_B knows A's long-term private key D_A and impersonates B to A in the attacking session by computing the session key $SK = H_2(\hat{e}(T_B, D_A)^x) = H_2(\hat{e}(T_A, D_A)^y)$ established at A.

A		E_B
	$\xrightarrow{T_A = xQ_B}$	
$SK = H_2(\hat{e}(T_B, D_A)^x)$	$\xleftarrow{T_B = yQ_B}$	$SK = H_2(\hat{e}(T_A, D_A)^y)$

Figure 5.2: Key Compromise Impersonation Attack on The MB-1 Protocol

To defeat the attack, Xie and McCullagh-Barreto attempted the following two variants respectively.

MB-2 (McCullagh-Barreto's second scheme (MB-2) [124]). A computes $K = \hat{e}(T_B, D_A) \cdot \hat{e}(P_1, P_2)^x = \hat{e}(P_1, P_2)^{x+y}$ and B computes $K = \hat{e}(T_A, D_B) \cdot \hat{e}(P_1, P_2)^y = \hat{e}(P_1, P_2)^{x+y}$. The agreed session key is $SK = H_2(\hat{e}(P_1, P_2)^{x+y})$. Although now the protocol achieves the key-compromise impersonation resilience property, this scheme loses another desirable security attribution: the perfect forward secrecy, i.e. with the knowledge of D_A and D_B, K can be computed by $K = \hat{e}(T_B, D_A) \cdot (T_A, D_B)$.

Xie's (Xie's scheme [163]). A computes $K = \hat{e}(T_B, D_A)^{x+1} \cdot \hat{e}(P_1, P_2)^x = \hat{e}(P_1, P_2)^{xy+x+y}$ and B computes $K = \hat{e}(T_A, D_B)^{y+1} \cdot \hat{e}(P_1, P_2)^y = \hat{e}(P_1, P_2)^{xy+x+y}$. The agreed session key is $SK = H_2(\hat{e}(P_1, P_2)^{xy+x+y})$. Unfortunately this modification is still vulnerable to the key-compromise impersonation attack [119].

A KCI attack on the protocol works as in Figure 5.3 where the adversary E_B impersonates B to A in the attacking session by computing the session key $SK = H_2(\hat{e}(T_B, D_A)^{x+1} \cdot \hat{e}(P_1, P_2)^x) = H_2(\hat{e}(\beta P_1 + sP_1, D_A)^{xy+y-1}) = H_2(\hat{e}(T_A, D_A)^y \cdot \hat{e}(P_1, P_2)^{y-1})$ established at A.

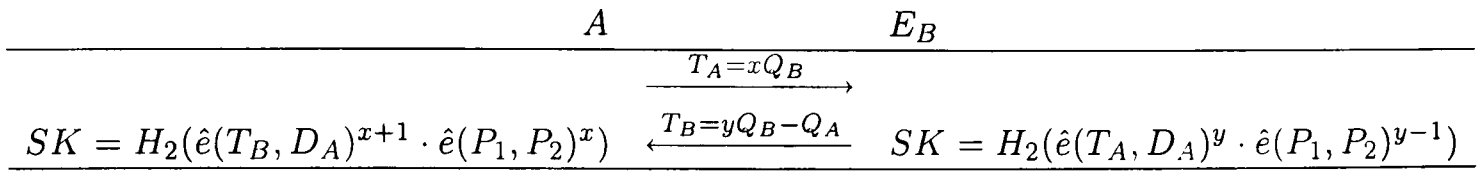

Figure 5.3: Key Compromise Impersonation Attack on The Xie Protocol

The protocol also suffers from a trivial man-in-the-middle attack [119] as in Figure 5.4 where the adversary $\mathcal{A}$ can compute $SK = \hat{e}(P_1, P_2)^{-1}$.

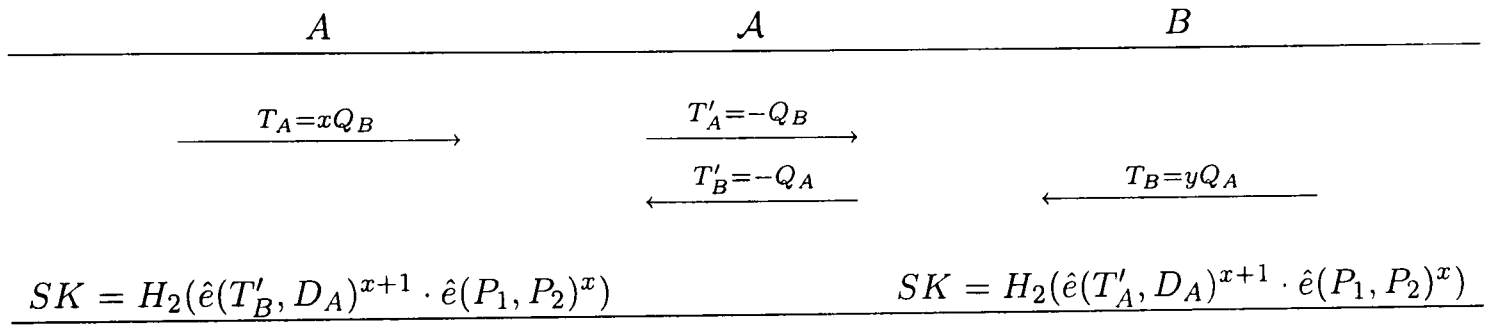

Figure 5.4: Man-In-The-Middle Attack on The Xie Protocol

5.5.2 On the Existing Security Proofs

A Sketch of Existing Proofs. Here we give a sketch of three security proofs from [125, 124, 163] respectively[2], each for one variant of the MB protocol as described in Section 5.5.1 implemented with Type-1 pairings, i.e. $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2 = P$. The proofs were intended to adopt a weakened Bellare-Rogaway key agreement model by forbidding all reveal queries.

All of the three proofs are based on the bilinear inverse Diffie-Hellman (BIDH) problem, described in Assumption 2.6.5, i.e. given $(P, \alpha P, \beta P)$, computing $\hat{e}(P, P)^{\beta/\alpha}$ is computationally infeasible. Each proof involves two algorithms: an adversary $\mathcal{A}$ and a challenger

[2]Xie's protocol suffers from the man-in-the-middle attack, therefore in principle it cannot be proven in the Bellare-Rogaway model, so there must be errors in the reduction. Here we still analyse the reduction and explicitly point out the errors, which again demonstrate the necessity of a valid security reduction of a scheme in a well-defined model to guarantee security.

(i.e. a simulator of the real world) $\mathcal{B}$. $\mathcal{A}$'s goal is to break a specified protocol, and $\mathcal{B}$'s goal is to solve the BIDH problem with the help of $\mathcal{A}$.

Each proof includes a set of parties modeled by oracles. $\mathcal{A}$ can access any oracle by issuing the queries of Create, Corrupt, Send, and Test. All queries by $\mathcal{A}$ pass through $\mathcal{B}$. Before the game starts, $\mathcal{B}$ randomly selects a pair of oracles, $\Pi_{i,j}^s$ and $\Pi_{j,i}^t$. $\mathcal{B}$ expects that $\mathcal{A}$ is going to attack the oracle $\Pi_{i,j}^s$ by playing the role of $\Pi_{j,i}^t$. In the three proofs, $\mathcal{A}$ and $\mathcal{B}$ play the game described in Section 5.2 in the following three slightly different ways.

Proof 1 (for MB-1 [125]). To answer a Create/Corrupt query for any oracle m where $m \neq j$, $\mathcal{B}$ chooses a random integer $y_m \in \mathbb{Z}_p^*$, and answers $y_m P$ as m's public key and $y_m^{-1} P$ as m's private key. $\mathcal{B}$ answers αP as j's public key and does not know j's private key $\alpha^{-1} P$. To answer a Send query for any oracle except $\Pi_{i,j}^s$, $\mathcal{B}$ follows the protocol properly. To answer a Send query for $\Pi_{i,j}^s$, $\mathcal{B}$ chooses a random integer $x_i \in \mathbb{Z}_p^*$ and answers $x_i P$. The proof relies on that an input from $\mathcal{A}$ as $\Pi_{j,i}^t$'s response is exactly the value of βP. After a Test query for $\Pi_{i,j}^s$, if $\mathcal{A}$ successfully breaks MB-1 by distinguishing the established key, K $= \hat{e}(P,P)^{x_i \beta / y_i \alpha}$, from a random number, $\mathcal{B}$ can get $\hat{e}(P,P)^{\beta/\alpha}$ by computing K^{y_i/x_i}.

Proof 2 (for MB-2 [124]). $\mathcal{B}$ answers Create/Corrupt queries in the same way as it did in Proof 1. To answer a Send query for any oracle except $\Pi_{i,j}^s$, $\mathcal{B}$ follows the protocol properly. To answer a Send query for $\Pi_{i,j}^s$, $\mathcal{B}$ answers βP. The input from $\mathcal{A}$ as $\Pi_{j,i}^t$'s response is an arbitrary value δP. After a Test query for $\Pi_{i,j}^s$, if $\mathcal{A}$ successfully breaks MB-2 by distinguishing the established key, $K = \hat{e}(P,P)^{\beta/\alpha + \delta/y_i}$, from a random number, $\mathcal{B}$ can get $\hat{e}(P,P)^{\beta/\alpha}$ by computing $K/\hat{e}(P,P)^{\delta/y_i}$.

Proof 3 (for Xie's protocol [163]). To answer a Create/Corrupt query for any oracle m where $m \notin \{i,j\}$, $\mathcal{B}$ chooses a random integer $y_m \in \mathbb{Z}_p^*$, and answers $y_m P$ as m's public key and $y_m^{-1} P$ as m's private key. $\mathcal{B}$ answers αP as i's public key and does not know i's private key $\alpha^{-1} P$. $\mathcal{B}$ answers βP as j's public key and does not know j's private key $\beta^{-1} P$. To answer a Send query for any oracle except $\Pi_{i,j}^s$, $\mathcal{B}$ follows the protocol properly.

To answer a Send query for $\Pi^s_{i,j}$, $\mathcal{B}$ chooses a random integer $x_i \in \mathbb{Z}^*_p$ and answers $x_i\beta P$. The proof relies on that an input from $\mathcal{A}$ as $\Pi^t_{j,i}$'s response is exactly the value of βP (i.e. $y_j\alpha P = \beta P$ for some y_j). After a Test query for $\Pi^s_{i,j}$, if $\mathcal{A}$ successfully breaks Xie's protocol by distinguishing the established key, $K = \hat{e}(P,P)^{(x_i+1)\beta/\alpha} \cdot \hat{e}(P,P)^{x_i}$, from a random number, $\mathcal{B}$ can get $\hat{e}(P,P)^{\beta/\alpha}$ by computing $(K/\hat{e}(P,P)^{x_i})^{1/(x_i+1)}$.

Analysis of Their Proofs. We now show that all the three reductions described in the last subsection are invalid. More specifically, the reductions have following three problems. *Problem 1: From $\mathcal{A}$'s point view, the simulation offered by $\mathcal{B}$ is distinguishable from the real world of an identity-based authenticated key agreement protocol.*

In any identity-based cryptographic system, the correctness of a private key derived from a chosen identity string, ID, is verifiable, given system public parameters. In those security reductions based on a standard model (e.g. [10, 12]), an adversary can use ID directly as the public key to verify the result of a private key generation query, i.e. Corrupt query. In those security reductions based on a random oracle model, such as [19, 60], to verify a correct key derivation can be done with a query of ID to the random oracle.

It is addressed in [125, 124, 163] that the identity map function H_1 in the MB protocol and its variants is by means of a random oracle. However, how to respond to the H_1 query is not specified in these three proofs. Another related missing part is that these three proofs do not specify either which entity has the access to the master secret key s, or what the system parameters that $\mathcal{A}$ would get access to should be. We can see that $\mathcal{B}$ is not able to answer the H_1 query by following the Create and Corrupt queries specified in these three proofs. As a result, $\mathcal{A}$ cannot verify correctness of either Create or Corrupt query result from ID and the system parameters. $\mathcal{A}$ can then immediately notice that $\mathcal{B}$ is a simulator, instead of the real world. We discuss this issue in the following two cases, dependent on whether or not $\mathcal{B}$ knows the value of s.

1. The value s is not known to $\mathcal{B}$. Following the three proofs, to answer the Create/Corrupt query to an oracle with the identity $\mathtt{ID}_m$, $\mathcal{B}$ assigns a random element pair, $y_m P$, $y_m^{-1} P \in \mathbb{G}_1^*$, as the public/private key pair. However, $\mathcal{B}$ is not able to give the value of $u_m = H_1(\mathtt{ID}_m)$, satisfying $u_m P = y_m P - sP$, because to solve the discrete logarithm problem in $\mathbb{G}_1$ is computationally infeasible, which is implied by the used BIDH assumption. Therefore, $\mathcal{B}$ is not able to answer the oracle query $H_1(\mathtt{ID}_m)$, and $\mathcal{A}$ then cannot verify correctness of the received $y_m P$ and $y_m^{-1} P$ from $\mathtt{ID}_m$ and sP.

2. The value s is chosen by $\mathcal{B}$. Following the reductions, to answer the Create query to an oracle with the identity $\mathtt{ID}_j$ in Proof 1 and Proof 2 (or $\mathtt{ID}_i$ in Proof 3), $\mathcal{B}$ assigns $\alpha P \in \mathbb{G}_1^*$, as the public key to the party j (or i). However, again $\mathcal{B}$ is not able to give the value of u_j (or u_i), satisfying $u_j P$ (or $u_i P$) $= \alpha P - sP$, because of the hardness of DLP in $\mathbb{G}_1$. Therefore $\mathcal{B}$ is not able to answer the oracle query $H_1(\mathtt{ID}_j) = u_j$ (or $H_1(\mathtt{ID}_i) = u_i$) and $\mathcal{A}$ then cannot verify correctness of the received public key αP for $\mathtt{ID}_j$ (or $\mathtt{ID}_i$) under the master public key sP.

In conclusion, since $\mathcal{B}$ cannot answer some H_1 queries, $\mathcal{A}$ can immediately notice the inconsistency between the simulation and the real world.

Problem 2: $\mathcal{B}$ in Proof 1 and Proof 3 requires that $\mathcal{A}$ provides an expected value as a response to a specific oracle. This is not a reasonable requirement.

The problem arises because $\mathcal{A}$ is not controlled by $\mathcal{B}$. Even the assumption that the adversary would follow the protocol strictly to generate messages is too strong to cover many dangerous attacks. A sound reduction can only require that messages from the adversary are in the specified message space at most.

Problem 3: H_2 is not clearly required in the MB protocol and its variants. As a result, the reduction to the computational BIDH assumption does not follow.

The simulation $\mathcal{B}$ cannot be created based on the BIDH assumption for the original

protocols in [125, 124, 163]. Instead, even if both Problem 1 and 2 are solved. a simulation could only be created based on the decision BIDH for the original protocols if they are secure. Otherwise, the adversary can win the game with the probability to differentiate a random element of $\mathbb{G}_t$ from the true value. This is the reason why we employ an extra hash function on the agreed secret to generate a session key SK.

5.5.3 A Modified Scheme and its Security Analysis

The MB-2 protocol does not achieve the known-session key security, i.e. the compromise of one session key would not affect other session keys' security, and so is insecure in the Bellare-Rogaway key agreement model with the *Reveal* query allowed.[3] The adversary $\mathcal{A}$ launches the known session key attack as in Figure 5.5, where $\mathcal{A}$ randomly chooses $r_{\mathcal{A}} \in \mathbb{Z}_p^*$ and issues the *Reveal* query to B's session of transcript (T'_A, T_B) to get the session key $SK = H_2(\hat{e}(P_1, P_2)^{x+y+r_{\mathcal{A}}})$. Then $\mathcal{A}$ chooses A's session of transcript (T_A, T'_B) as the fresh oracle to issue the *Test* query. The session of A and B have different transcripts, therefore $\mathcal{A}$ is allowed to issue the *Test* query on the chosen session which is fresh. Because the two sessions have established the same session key, $\mathcal{A}$ wins the game trivially.

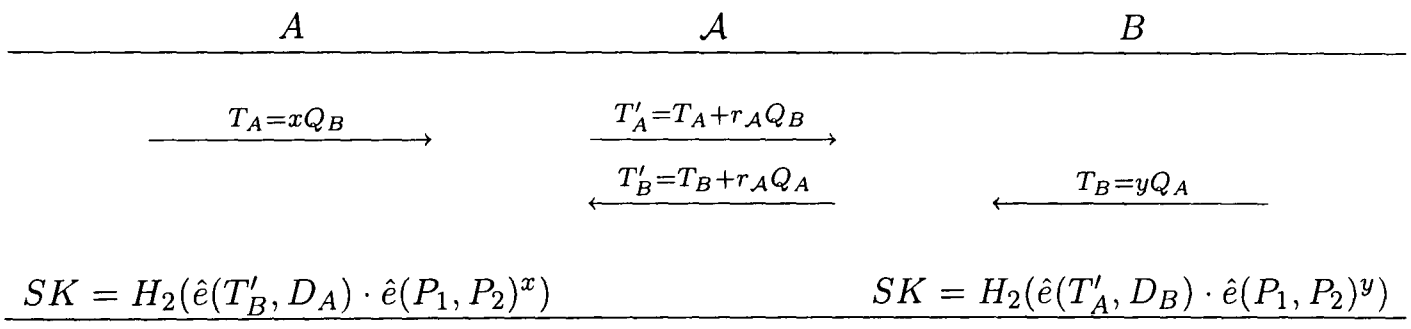

Figure 5.5: Known Session Key Attack on The MB-2 Protocol

We can slightly tweak the session key generation method in MB-2 to recover from the attack, furthermore the modification enables us to reduce the security of the scheme to the

[3]Note that the reductions in [124, 125] proceed in the model with the *Reveal* query disallowed, though they are invalid.

ℓ-GBCAA1 assumption in the full model presented in Section 5.5.2. The modified scheme, which we refer to as **MB-2'**, generates the session key as follows:

$$SK = H_3(A, B, T_A, T_B, \hat{e}(P_1, P_2)^{x+y}),$$

where $H_3 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_t \rightarrow \{0,1\}^n$ is a hash function. As the original protocol, the new scheme still cannot achieve PFS.

Theorem 5.5.1. *MB-2' is a secure AK, provided that H_1, H_3 are random oracles and the ℓ-GBCAA1 assumption is sound. Specifically, suppose that there is a PPT adversary $\mathcal{A}$ against the protocol with non-negligible probability $\epsilon(k)$ and in the attack H_1 has been queried q_1 times and q_o oracles have been created. Then there exists a PPT algorithm $\mathcal{B}$ to solve the $(q_1$-1)-GBCAA$1_{1,2}$ problem with advantage*

$$Adv_{\mathcal{B}}^{(q_1-1)-GBCAA1_{1,2}}(k) \geq \frac{1}{q_1 \cdot q_o}\epsilon(k).$$

Proof: Condition 1 of Definition 5.2.2 directly follows from the protocol specification. In the sequel we prove that the protocol satisfies Condition 2. We show that if $\mathcal{A}$ exists, we can construct an algorithm $\mathcal{B}$ to solve a $(q_1 - 1)$-GBCAA$1_{1,2}$ problem with non-negligible probability.

Given an instance of the (q_1-1)-GBCAA$1_{1,2}$ problem $(sP_1, h_0, (h_1, \frac{1}{h_1+s}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2))$ with a set of pairing parameter where $h_i \in_R \mathbb{Z}_p^*$ for $0 \leq i \leq q_1 - 1$ and the DBIDH$_{1,1}$ oracle $\mathcal{O}_{DBIDH}$, $\mathcal{B}$ simulates the **Setup** algorithm to generate the system parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, P_1, P_2, sP_1, H_1, H_3)$, i.e. using s as the master secret key which it does not know. H_1 and H_3 are two random oracles controlled by $\mathcal{B}$.

$\mathcal{B}$ randomly chooses $1 \leq I \leq q_1$ and $1 \leq J \leq q_o$, and interacts with $\mathcal{A}$ in the following way:

- $H_1(\mathtt{ID}_i)$: $\mathcal{B}$ maintains a list of tuples $(\mathtt{ID}_i, h_i, D_i)$ as explained below. We refer to this list as H_1^{list}. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point on $\mathtt{ID}_i$, $\mathcal{B}$ responds as follows:
  - If $\mathtt{ID}_i$ already appears on the H_1^{list} in a tuple $(\mathtt{ID}_i, h_i, D_i)$, then $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = h_i$.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $(\mathtt{ID}_I, h_0, \perp)$ into the tuple list and responds with $H_1(\mathtt{ID}_I) = h_0$.
  - Otherwise, $\mathcal{B}$ selects a random integer $h_i (i > 0)$ from the $(q_1 - 1)$-GBCAA$1_{1,2}$ instance which has not been chosen by $\mathcal{B}$ and stores $(\mathtt{ID}_i, h_i, \frac{1}{h_i+s}P_2)$ into the tuple list. $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = h_i$.

- $H_3(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, K_t)$: At any time $\mathcal{A}$ can issue queries to the random oracle H_3. To respond to these queries, $\mathcal{B}$ maintains a list of tuples called H_3^{list}. Each entry in the list is a tuple of the form $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, K_t, \zeta_t, O_t)$ indexed by $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, K_t)$. To respond to a query, $\mathcal{B}$ does the following operations:
  - If on the list there is a tuple indexed by $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, K_t)$, then $\mathcal{B}$ responds with ζ_t.
  - Otherwise, $\mathcal{B}$ goes through the list Λ built in the *Reveal* query to find tuples of the form $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, r^t, \zeta^t, O^t)$ and proceeds as follows:
    * Compute $D = K_t/(P_1, P_2)^{r^t}$.
    * Access $\mathcal{O}_{DBIDH}((h_0+s)P_1, T_j, D)$ if $O^t = 0$ or access $\mathcal{O}_{DBIDH}((h_0+s)P_1, T_i, D)$ otherwise. If $\mathcal{O}_{DBIDH}$ returns 1, $\mathcal{B}$ inserts $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, K_t, \zeta^t)$ into H_3^{list} and responds with ζ^t to the query and removes the tuple from Λ.
  - Otherwise, $\mathcal{B}$ randomly chooses a string $\zeta_t \in \{0,1\}^n$ and inserts a new tuple $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, K_t, \zeta_t)$ into the list H_3^{list}. It responds to $\mathcal{A}$ with ζ_t.
- Corrupt($\mathtt{ID}_i$): $\mathcal{B}$ looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. $\mathcal{B}$ checks the value of D_i: if $D_i \neq \perp$, then $\mathcal{B}$ responds with D_i; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).
- Send($\Pi_{i,j}^t, T$): $\mathcal{B}$ maintains a list with tuples of $(\Pi_{i,j}^t, r_{i,j}^t, tran_{i,j}^t)$ and responds to the query as follows:
  - $\mathcal{B}$ looks through the list H_1^{list}. If $\mathtt{ID}_j$ is not on the list, $\mathcal{B}$ queries $H_1(\mathtt{ID}_j)$. After that, $\mathcal{B}$ checks the value of t.
  - If $t \neq J$, $\mathcal{B}$ honestly follows the protocol to respond the query by randomly sampling $r_{i,j}^t \in \mathbb{Z}_p^*$ and generating the message $r_{i,j}^t(H_1(\mathtt{ID}_j)P_1 + sP_1)$.
  - If $t = J$, $\mathcal{B}$ further checks the value of D_j, and then responds the query differently as below depending on this value.
    * If $D_j \neq \perp$, $\mathcal{B}$ aborts the game (**Event 2**). We note that there is only one party's private key is represented as $\perp$ in the whole simulation.
    * Otherwise, $\mathcal{B}$ randomly chooses $y \in \mathbb{Z}_p^*$ and responds with yP_1. Note that $\Pi_{i,j}^t$ can be the initiator (if $T = \lambda$) or the responder (if $T \neq \lambda$).
- Reveal($\Pi_{i,j}^t$): $\mathcal{B}$ maintains an initially empty list Λ with tuples $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, r^t, \zeta^t, O^t)$. $\mathcal{B}$ responds to the query as follows:
  - If $t = J$ or if the J-th oracle has been generated as $\Pi_{a,b}^J$ and $\mathtt{ID}_a = \mathtt{ID}_j, \mathtt{ID}_b = \mathtt{ID}_i$ and two oracles have the same session ID, then abort the game (**Event 3**).
  - Go through $H_1^{list}(\mathtt{ID}_i)$ to find the private key D_i of party i with identity $\mathtt{ID}_i$.
  - If $D_i \neq \perp$, compute $K = \hat{e}(T_i, D_i) \cdot \hat{e}(P_1, P_2)^{r_{i,j}^t}$ where T_i is the incoming message and $r_{i,j}^t$ is the random flips of the oracle $\Pi_{i,j}^t$. $\mathcal{B}$ responds with $H_3(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, K)$ if the oracle is the initiator, otherwise $H_3(\mathtt{ID}_j, \mathtt{ID}_i, T_j, T_i, K)$.

- Otherwise, $\mathcal{B}$ goes through H_3^{list} to find tuples indexed by $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j)$ (if $\Pi_{i,j}^t$ is the initiator) or by $(\mathtt{ID}_j, \mathtt{ID}_i, T_j, T_i)$ (if $\Pi_{i,j}^t$ is the responder). For each (K_t, ζ_t) in the found tuples,
  * Compute $D = K_t / \hat{e}(P_1, P_2)^{r_{i,j}^t}$.
  * Access $\mathcal{O}_{DBIDH}((h_0 + s)P_1, T_i, D)$. If $\mathcal{O}_{DBIDH}$ returns 1, then $\mathcal{B}$ responds to the query with ζ_t. Note that there can be at most one K_t meeting the test.
- Otherwise (no K_t is found in the last step), randomly choose $\zeta^t \in \{0,1\}^n$ and insert $(\mathtt{ID}_i, \mathtt{ID}_j, T_i, T_j, r_{i,j}^t, \zeta^t, 1)$ if the oracle is the initiator, or $(\mathtt{ID}_j, \mathtt{ID}_i, T_j, T_i, r_{i,j}^t, \zeta^t, 0)$ into Λ. $\mathcal{B}$ responds with ζ^t.

- Test($\Pi_{i,j}^t$): If $t \neq J$ or ($t = J$ but) there is an oracle $\Pi_{j,i}^w$ with the same session ID with $\Pi_{i,j}^t$ that has been revealed, $\mathcal{B}$ aborts the game (**Event 4**). Otherwise, $\mathcal{B}$ randomly chooses a number $\zeta \in \{0,1\}^n$ and gives it to $\mathcal{A}$ as the response.

Once $\mathcal{A}$ finishes the queries and returns its guess, $\mathcal{B}$ goes through H_3^{list} and for each K_ξ,

- Compute $D = (K_\xi / \hat{e}(T, D_i))^{1/y}$, where T is the incoming message to the tested oracle $\Pi_{i,j}^J$.
- Access $\mathcal{O}_{DBIDH}((h_0+s)P_1, P_1, D)$. If $\mathcal{O}_{DBIDH}$ returns 1, $\mathcal{B}$ returns D as the response of the $(q_1 - 1)$-GBCAA$1_{1,2}$ challenge.
- If no K_ξ meets the test, fail the game.

Claim 5.5.1. *If algorithm $\mathcal{B}$ does not abort during the simulation, then algorithm $\mathcal{A}$'s view is identical to its view in the real attack.*

Proof: $\mathcal{B}$'s responses to H_1 queries are uniformly and independently distributed in $\mathbb{Z}_p^*$ as in the real attack. H_3 is modeled as a random oracle which requires that for each unique input, there should be only one response. We note that the simulation substantially makes use of the programmability of random oracle and the access to the DBIDH oracle to guarantee the unique response for every H_3 query. The responses in other types of query are valid as well. Hence the claim follows.

Note the agreed secret in the chosen fresh oracle $\Pi_{i,j}^t$ should be $K = \hat{e}(T, D_i) \cdot \hat{e}(P_1, P_2)^r$ where $r(h_0 P_1 + s P_1) = y P_1$ (recall that party j's public key is $h_0 P_1 + s P_1$ and the private key is unknown to $\mathcal{B}$ and represented by $\perp$), i.e. $r = \frac{y}{h_0+s}$ and $K = \hat{e}(T, D_i) \cdot \hat{e}(yP_1, \frac{1}{h_0+s} P_2)$. Following the similar argument as in Claim 5.4.2, we have following claim.

Claim 5.5.2. *Let **Event 5** be that $K = \hat{e}(T, D_i) \cdot \hat{e}(yP_1, \frac{1}{h_0+s} P_2)$ was not queried on H_3. Then* $\Pr[\overline{Event\ 5}] \geq \epsilon(k)$.

Let **Event 6** be that, in the attack, adversary B indeed chose oracle $\Pi_{i,j}^J$ as the challenger oracle where $\mathtt{ID}_j$ was queried on H_1 as the I-th distinct identifier query. Then following the

rules of the game defined in Section 5.2, it's clear that **Event 1, 2, 3, 4** would not happen. So,

$$\Pr[\overline{(\texttt{Event 1} \vee \texttt{Event 2} \vee \texttt{Event 3} \vee \texttt{Event 4})}] = \Pr[\texttt{Event 6}] \geq \frac{1}{q_1 \cdot q_o}.$$

Overall, we have

$$\begin{aligned} \Pr[A \text{ wins}] &= \Pr[\texttt{Event 6} \wedge \overline{\texttt{Event 5}}] \\ &\geq \frac{1}{q_1 \cdot q_o}\epsilon(k). \end{aligned}$$

This completes the security analysis of the protocol. □

If one considers that the used assumption for MB-2′ is too strong, another variant is to instantiate the SIG-DH [59] with the BLMQ-IBS [25] as the signature scheme. The protocol can be proven based on the ℓ-SDH assumption [59], but is relatively slower than MB-2′. In the protocol, each party needs to compute one pairing, four multiplications in $\mathbb{G}_1$ and two exponentiations in $\mathbb{G}_t$, while the MB-2′ protocols requires one pairing, two multiplications in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_t$.

5.6 An Identity-Based KAP with Unilateral Identity Privacy

In previous sections, we have discussed some general IB-KAPs. These protocols more or less achieve the general security properties such as the known-session key security, KCI-resilience, etc. While, apart from these general security properties, some other special security properties might also be required in certain environments. User *Identity Privacy (IP)*, which means that no outsider adversary can determine the participant's identity in each protocol execution, is of particular interest in some environments.

For some network services, the service subscriber's identity is a piece of sensitive information which must be protected from outsiders. A typical example of such service would be the mobile network access. In the mobile network, the mobile station as a client and the base station (or access point) as a server have to authenticate each other, so the mobile station can be assured that it is using the authentic service provider, instead of a bogus access network; while by authenticating the mobile station, the base station can control

the access to the network service. For the nature of the wireless communication, the signals can be easily eavesdropped. If the mobile station's identifier is not protected during the authentication process, an adversary can easily obtain this information. By using the identifier, an adversary can further obtain other information which could be valuable to users. The adversary may locate a user and track his or her movement; also it can analyse the traffic between the user and the network to extract the user's communication behavior. Hence, the privacy of user identity over the air interface in mobile networks is essential to guarantee the user's location privacy and prevent an adversary from associating a user with his or her activities. Another example would be that the service itself is sensitive, such as the crime report hotline. The user would not want others to know who has accessed such a service. Even in general network environments, some key agreement protocols such as the Internet Key Exchange protocol [95], also regard identity privacy as a desired property.

Achieving identity privacy in a key agreement protocol is not a new challenge. Many efforts were made to address this issue, especially in the mobile networks, e.g. [27, 1, 96]. However, these schemes rely on either the symmetric-key cryptography or the traditional asymmetric-key cryptography so that they may face various challenges, such as secret key sharing or certification distribution, in applications. In this section we propose an efficient identity-based two-party key agreement protocol with client identity privacy for use in the client-server environment, and investigate its security properties.

5.6.1 Description of the Scheme

The scheme adopts the basic SOK key construction method. In the system, the KGC generates the system parameters and private keys via the following Setup and Extract algorithm respectively:

Setup. The KGC executes the following operations:

1. On input 1^k, generates a set of pairing parameters of the required size.

2. Pick a random $s \in \mathbb{Z}_p$ and compute $R = sP_2$.

3. Pick four cryptographic hash functions as follows:

$$H_1 : \{0,1\}^l \rightarrow \mathbb{G}_2.$$
$$H_2 : \mathbb{G}_2 \times \mathbb{G}_2 \rightarrow \mathbb{Z}_p.$$
$$H_3 : \mathbb{G}_t \rightarrow \{0,1\}^w.$$
$$H_4 : \{0,1\}^l \times \{0,1\}^l \times \mathbb{G}_2 \times \mathbb{G}_2 \times \{0,1\}^w \times \mathbb{G}_t \rightarrow \{0,1\}^n$$

   for some integer $n, l > 0$ and $w = \lceil \log_2 p \rceil + l$ ($\lceil \log_2 p \rceil$ is the bit length of p).

4. Set s as the master secret key which is kept secret by KGC and publish others as the system public parameters (also called the master public key).

Extract. For any user with an identity $\mathtt{ID}_A \in \{0,1\}^l$, given the master secret key s and the system public parameters, the KGC executes the SOK **Extract** algorithm to compute $Q_A = H_1(\mathtt{ID}_A)$, $D_A = sQ_A \in \mathbb{G}_2$ and passes D_A as the private key to this user via some secure channel.

We suppose that the client possesses the identity A and the server possesses the identity B. Without loss of generality, we assume that there is always an association procedure (such as a TCP connection procedure or a physical connection establishment phase) between the client and the server before starting the key agreement protocol. Once the association has been established, A and B proceed as follows (this process is also depicted in Figure 5.6).

1. The server B randomly chooses $r_B \in \mathbb{Z}_p$ and sends $(B, r_B Q_B)$ to the client, where $Q_B = H_1(B)$. It should be noted that the server B does not know the identity of the client.

2. The client A responds as follows: 1) randomly choose $r_A \in \mathbb{Z}_p$; 2) compute $h = H_2(r_A Q_A, r_B Q_B)$; 3) compute the agreed secret $K = \hat{e}(\psi(D_A), r_B Q_B + hQ_B)^{r_A}$; 4) generate a mask $Z = (r_A \| A) \oplus H_3(K)$; 5) send $r_A Q_A$ and the mask Z to the server B.

$(Client)A$		$B(Server)$
$Q_A = H_1(A)$		$Q_B = H_1(B)$
	$\xleftarrow{B, r_B Q_B}$	$r_B \in_R \mathbb{Z}_p$
$r_A \in_R \mathbb{Z}_p$;		
$h = H_2(r_A Q_A, r_B Q_B)$;		
$K = \hat{e}(\psi(D_A), r_B Q_B + h Q_B)^{r_A}$;		
$Z = (r_A \| A) \oplus H_3(K)$	$\xrightarrow{r_A Q_A, Z}$	$h = H_2(r_A Q_A, r_B Q_B)$;
		$K = \hat{e}(\psi(r_A Q_A), D_B)^{r_B + h}$;
		$r_C \| C = Z \oplus H_3(K)$;
		$r_C H_1(C) \stackrel{?}{=} r_A Q_A$
$SK = H_4(A, B, r_A Q_A, r_B Q_B, Z, K)$		$SK = H_4(C, B, r_A Q_A, r_B Q_B, Z, K)$

Figure 5.6: Identity-based Key Agreement with Unilateral Identity Privacy

3. The server B processes the incoming message as follows: 1) compute $h = H_2(r_A Q_A, r_B Q_B)$; 2) compute the agreed secret $K = \hat{e}(\psi(r_A Q_A), D_B)^{r_B + h}$; 3) recover the client identifier C and the client's random flips r_C from the mask Z by $r_C \| C = Z \oplus H_3(K)$; 4) check if the equation $r_C H_1(C) = r_A Q_A$ holds. If the equation does not hold, B aborts the protocol.

4. On completion of the AK protocol, both parties compute the session key SK as specified in Figure 5.6.

We note that the protocol does not have to work in the client/server mode. Two peer parties can also establish session keys with or without protecting the responder's identity privacy. The protocol specification as above works in both situations. However, because so, the protocol specification requires a hash function H_1 whose codomain is $\mathbb{G}_2$ and a homomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$. Hence this protocol specification can only be implemented with Type-1 and Type-4 pairings. When the protocol is implemented with Type-4 pairings, it can be optimised by replacing $r_B Q_B$ with $r_B \psi(Q_B)$ and then A computes

$K = \hat{e}(r_B\psi(Q_B) + h\psi(Q_B), D_A)^{r_A}$ and B computes $K = \hat{e}(\psi(D_B), r_AQ_A)^{r_B+h}$, or by replacing r_AQ_A with $r_A\psi(Q_A)$ and then B checks $r_C\psi(H_1(C)) \stackrel{?}{=} r_A\psi(Q_A)$. If the check on the message membership is carried out, the former optimisation is faster.

If the protocol works only in the client/server mode, the pairing-type restriction can be overcomed as follows: Choose the following two hash functions to replace H_1,

$$H_1' : \{0,1\}^* \rightarrow \mathbb{G}_2.$$
$$H_1'' : \{0,1\}^l \rightarrow \mathbb{G}_1.$$

The server identity $\mathtt{ID}_S$ is mapped to an element in $\mathbb{G}_2$ by $Q_S = H_1'(\mathtt{ID}_S)$ and the client identity $\mathtt{ID}_C$ is mapped to an element in $\mathbb{G}_1$ by $Q_C = H_1''(\mathtt{ID}_C)$. And ψ is the trivial identity map. We can also switch the codomain of H_1' and H_1'' and the inputs of pairing computation. The domain of H_2 and H_3 should be adjusted correspondingly.

5.6.2 Security Model of KAP with Identity Privacy

The model defined in Section 5.2 cannot be directly used to formulate the protocols with the identity privacy property. In the protocol with identity privacy, a party may not know its partner's identity before some stage of the protocol. Hence in the formulation if an oracle does not know its partner's identifier, it cannot use this information when generating the response, while at the end of the protocol, the oracle should be able to output the intended partner's identifier, if it accepts the session.

There is an obstacle in the Bellare-Rogaway model to simulate the situation that an adversary only passively eavesdrops the conversation between two parties when the protocol has identity privacy, because in the model, the adversary supposes to fully control the network and to schedule the messages among the parties. Without knowing the identity of oracles, the adversary cannot dispatch the messages. In [61], Canetti and Krawczyk introduced an extra identifier "destination address" to simulate the "post-specified peer setting" which is a related notion to identity privacy. In the model of [61], a session is activated by the adversary using a triple (i, t, d) where i is the party identifier, t is the

session identifier, and d is the destination address. However, although using destination address d can simulate the situation that the party sends out a message to an unknown party with address d, such activation still allows the adversary to know the source identifier of the message.

Instead of introducing further complexity into the Bellare-Rogaway model, we consider identity privacy only when simulating the behavior of an honest party who strictly follows the protocol specification and is modeled as an oracle, and allow the adversary to know the source and possibly destination identifiers of each message generated by these honest parties. In other words, we can think that in the attack the honest parties reveal to the adversary the hidden identifiers in messages. While, the adversary is not required to know the identifiers in a message faked by itself. Hence the adversary's ability is not restricted, but the model is only used to check the common security attributes, such as mutual key authentication, known session key security, etc. Identity privacy has to be scrutinised separately.

To use the Bellare-Rogaway model, we have to make some necessary changes. (1) $\Pi^s_{i,*}$ is used to denote an oracle. Note that the pair (i, s) is able to uniquely identify an oracle simulating a party i in its s-th session. (2) The queries defined in Section 5.2 should be adopted for Change 1.

- $Send(\Pi^s_{i,*}, x)$. Oracle $\Pi^s_{i,*}$ who does not know its partner so far, follows the protocol $\Pi(1^k, i, S_i, P_i, tran^s_{i,*}, r^s_{i,*}, x)$ to generate response. We note that the partner's identifier and public key are not used as the input to algorithm Π, hence this query is not just about changing of the notation of an oracle from $\Pi^s_{i,j}$ to $\Pi^s_{i,*}$, but restricting the behavior of the oracles. If the oracle recovers a partner identifier j through some algorithm $\mathcal{F}$: $j = \mathcal{F}(1^k, i, S_i, P_i, tran^s_{i,*}, r^s_{i,*}, x)$, it replaces the unknown partner identifier $*$ with j and retrieves j's public key P_j. This is a new query in the model, but a similar formulation has been used in [61].

- *Reveal*($\Pi^s_{i,*}$). If the oracle has not accepted, it returns $\perp$; otherwise, it must have known its partner j, and then $\Pi^s_{i,j}$ reveals the session key. Here, the oracle is not required to disclose the partner's identity even if it has accepted the session (we note that the adversary may not know the oracle's partner because the message to $\Pi^s_{i,*}$ may be faked by the adversary).

We further note that an oracle $\Pi^s_{i,*}$ to be tested should be fresh, hence it has accepted and known its partner j.

5.6.3 Security Analysis of the Scheme

Before formally analysing the protocol, we describe a game to ease the analysis.

Interactive game with a BDH challenger. An algorithm $\mathcal{B}$ with a pair of PPT sub-algorithms $(\mathcal{B}_1(r_1;\ \cdots), \mathcal{B}_2(r_2;\ \cdots))$ where r_i is used by $\mathcal{B}_i$ as the random tape, engages with a challenger in the following game:

Interactive $\mathrm{BDH}_{i,j,k}$ game

$(aP_i, bP_j, cP_k) \leftarrow \mathcal{F}(1^t)$;
$(X, \sigma) \leftarrow \mathcal{B}_1(r_1;\ aP_i, bP_j, cP_k)$;
$h \leftarrow \mathbb{Z}_p$;
$K \leftarrow \mathcal{B}_2(r_2;\ h, \sigma)$.

where $\mathcal{F}$ is a BDH challenge generator with parameters 1^t; $i, j, k \in \{1, 2\}$; $a, b, c \in_R \mathbb{Z}_p$; σ is the state information passed from $\mathcal{B}_1$ to $\mathcal{B}_2$ and $X \in \mathbb{G}_2$. We say that $\mathcal{B}$ wins the game if it computes $K = \hat{e}(P_1, X + hbP_2)^{ac}$. Define the advantage of the adversary as the function of t by

$$\mathrm{Adv}_{\mathcal{B}}(t) = \Pr[\mathcal{B} \text{ wins}].$$

Theorem 5.6.1. *If the $BDH_{i,j,k}$ assumption is sound, any PPT algorithm $\mathcal{B}$ participating in the interactive $BDH_{i,j,k}$ game can only have negligible advantage.*

Proof: Given a $\text{BDH}_{i,j,k}$ instance (aP_i, bP_j, cP_k), we construct an algorithm in the following way:

$(X, \sigma) \leftarrow \mathcal{B}_1(r_1;\ aP_i, bP_j, cP_k);$
$h \leftarrow \mathbb{Z}_p;$
$K_1 \leftarrow \mathcal{B}_2(r_2;\ h, \sigma);$
$\triangleright_1$
$K_2 \leftarrow \mathcal{B}_2(r_2;\ h-1, \sigma);$
$\triangleright_2$
output K_1/K_2.

Claim 5.6.1. *For any algorithm $\mathcal{B}$ with PPT sub-algorithms $(\mathcal{B}_1, \mathcal{B}_2)$ with advantage $\epsilon(t)$ to win the interactive BDH game, the above algorithm outputs $\hat{e}(P_1, P_2)^{abc}$ with probability $\epsilon^2(t)$.*

Proof: At point $\triangleright_1$, the algorithm computes correct $K_1 = \hat{e}(P_1, X + hbP_2)^{ac}$ with probability $\epsilon(t)$. Because h is chosen independently from X and σ, two execution of $\mathcal{B}_2$ should have the equal probability to output the correct answer. Hence at point $\triangleright_2$, the event that $K_1 = \hat{e}(aP, X + hbP)^c$ and at the same time $K_2 = \hat{e}(P_1, X + (h-1)bP_2)^{ac}$ happens with probability $\epsilon^2(t)$. The claim follows.

The theorem follows from the claim. □

Now, we investigate the security of the proposed protocol.

As defined in Definition 5.2.1, for a chosen fresh oracle $\Pi^s_{i,j}$ in the game, party i can be corrupted, which is particularly used to address the known-key impersonation resilience property. Because the protocol differentiates the role of parties, i.e. some are clients and others are servers, and this can be done by requiring that the client and the server identifers are in two separate sets, say $(\texttt{ID}^C, \texttt{ID}^S)$, we consider the security of the AK protocol in the following two cases:

(1) *Server authentication.* We consider the known-(client)-key attack that the adversary tries to impersonate a server to a client whose private key is known to the adversary or the adversary is benign (honestly conveying messages between oracles), i.e. the chosen fresh oracle $\Pi^t_{i,j}$ is with $i \in \texttt{ID}^C$ and $j \in \texttt{ID}^S$ and i is corrupted by the adversary.

Lemma 5.6.2. *The protocol is secure against the known-client-key attack, provided the $BDH_{2,2,2}$ game is hard to win and the hash functions are modeled as random oracles. Specifically, suppose there is a known-client-key PPT adversary $\mathcal{A}$ against the protocol with non-negligible probability $\epsilon(k)$ and in the attack q_1* server *identifiers are queried on H_1 and*

adversary $\mathcal{A}$ creates q_C client *oracles and queries q_3 and q_4 times on H_3 and H_4 respectively. Then there exists a PPT algorithm $\mathcal{B}$ to win the $BDH_{2,2,2}$ game with advantage*

$$Adv_{\mathcal{B}}^{BDH_{2,2,2}}(k) \geq [\frac{\epsilon(k)}{q_1 \cdot q_C \cdot \max\{q_3, q_4\}}]^2.$$

Proof: The first condition is trivial to prove. Now we prove that the protocol meets the second condition.

We prove the lemma by constructing an algorithm $\mathcal{B}$ using the adversary $\mathcal{A}$ against the protocol as a subroutine to win the interactive $\mathrm{BDH}_{2,2,2}$ game with non-negligible advantage. The game proceeds in the following way. After $\mathcal{G}$ outputs the challenge (sP_2, aP_2, bP_2), $\mathcal{B}$ simulates the system setup to adversary $\mathcal{A}$ as follows. The system parameters are set as $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, sP_2, H_1, H_2, H_3, H_4)$, i.e. the master secret key is s, which $\mathcal{B}$ does not know; H_1, H_2, H_3, H_4 are random oracles controlled by $\mathcal{B}$.

Recall that in the protocol, it is required that the client and server identifers are in two separate sets $(\mathtt{ID}^C, \mathtt{ID}^S)$, which is the way to differentiate the roles of parties. In this proof, we use two types of **Send** query to reflect the behavior of the protocol.

- **Send**$^S(\Pi_{i,*}^t,$ **M**): This query is sending a message to a server oracle. Message M is in the form of (Y, Z) or λ. If $M = \lambda$, a server oracle will be created. We use $*$ to represent the sender because the receiver of M does not know the sender before it unmasks the message in the real world. In the simulation, $\mathcal{B}$ cannot make use of knowledge of its partner's identifier before it decrypts the incoming message.
- **Send**$^C(\Pi_{i,*}^t,$ **M**): This query is sending a message to a client oracle. Message M is in the form of (j, X). This query will trigger the creation of a client oracle. As in the previous proofs in this chapter, we slightly abuse the notation. We use $\Pi_{i,*}^t$ to denote the t-th client oracle among all the client oracles created in the game, instead of the t-th instance of party i.

Algorithm $\mathcal{B}$ randomly chooses $1 \leq I \leq q_1$ and $1 \leq J \leq q_C$ and starts simulating the real world where the adversary $\mathcal{A}$ launches the attack by answering queries as follows.

- $H_1(\mathtt{ID}_i)$: $\mathcal{B}$ maintains an initially empty list H_1^{list} with entries of the form $(\mathtt{ID}_i, Q_i, \ell_i)$. $\mathcal{B}$ responds to the query in the following way.
  - If $\mathtt{ID}_i$ already appears on H_1^{list} in a tuple $(\mathtt{ID}_i, Q_i, \ell_i)$, then $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = Q_i$.
  - If $\mathtt{ID}_i$ is the I-th unique *server* identifier query, then $\mathcal{B}$ inserts $(\mathtt{ID}_i, bP_2, \perp)$ into the list and returns bP.
  - Otherwise, $\mathcal{B}$ randomly chooses $\ell_i \in \mathbb{Z}_p$, inserts $(\mathtt{ID}_i, \ell_i P_2, \ell_i)$ into the list and returns $\ell_i P_2$.
- $H_2(X_i, Y_i)$: $\mathcal{B}$ maintains an initially empty list H_2^{list} with entries of the form (X_i, Y_i, z_i) indexed by (X_i, Y_i). $\mathcal{B}$ responds to the query in the following way.

  - If a tuple indexed by (X_i, Y_i) is on the list, then $\mathcal{B}$ responds with z_i.
  - Otherwise, $\mathcal{B}$ randomly chooses $z_i \in \mathbb{Z}_p$, inserts (X_i, Y_i, z_i) into the list and returns z_i.

- $H_3(K_i)$: $\mathcal{B}$ maintains an initially empty list H_3^{list} with entries of the form (K_i, k_i) indexed by K_i. $\mathcal{B}$ responds to the query in the following way.
  - If a tuple indexed by K_i is on the list, then $\mathcal{B}$ responds with k_i.
  - Otherwise, $\mathcal{B}$ randomly chooses $k_i \in \{0,1\}^w$, inserts (K_i, k_i) into the list and returns k_i.

- $H_4(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, Z_i, K_i)$: $\mathcal{B}$ maintains an initially empty list H_4^{list} with entries of the form $(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, Z_i, K_i, \zeta_i)$ indexed by $(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, Z_i, K_i)$. $\mathcal{B}$ responds to the query in the following way.
  - If a tuple indexed by $(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, Z_i, K_i)$ is on the list, then $\mathcal{B}$ responds with ζ_i.
  - Otherwise, $\mathcal{B}$ randomly chooses $\zeta_i \in \{0,1\}^n$, inserts $(\mathtt{ID}_i, \mathtt{ID}_j, X_i, Y_j, Z_i, K_i, \zeta_i)$ into the list and returns ζ_i.

- **Corrupt**$(\mathtt{ID}_i)$: $\mathcal{B}$ looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. $\mathcal{B}$ checks the value of ℓ_i: if $\ell_i \neq \perp$, then $\mathcal{B}$ responds with $\ell_i sP$; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).

- **Send**$^S(\Pi_{i,*}^t$, **M**): $\mathcal{B}$ maintains a list Ω for every (client or server) oracle of the form $(\Pi_{i,j}^t, r_{i,j}^t, tran_{i,j}^t, K_{i,j}^t, SK_{i,j}^t)$ where $r_{i,j}^t$ is the random string used to generate message; $tran_{i,j}^t$ is the transcript of the oracle so far, and $K_{i,j}^t$ and $SK_{i,j}^t$ are set $\perp$ initially. $\mathcal{B}$ proceeds in the following way:
  - If $M = \lambda$, $\mathcal{B}$ randomly chooses $r \in \mathbb{Z}_p$ and responds with (i, rQ_i) where Q_i is found from H_1^{list} with identifier i (i.e. $r_{i,*}^t = r$).
  - Otherwise ($M = (Y, Z)$), $\mathcal{B}$ proceeds in the following way:
    * If ℓ_i on H_1^{list} corresponding to i is not $\perp$, then compute $K = \hat{e}(\psi(Y), \ell_i sP_2)^{r_{i,*}^t + z}$ where $z = H_2(Y, r_{i,*}^t Q_i)$. Use $MK = H_3(K)$ to unmask Z to recover (r_C, C). If $Y = r_C H_1(C)$, then set the partner of the oracle as C, $K_{i,C}^t = K$, and $SK_{i,C}^t = H_4(C, i, Y, r_{i,C}^t Q_i, Z, K_{i,C}^t)$; otherwise, reject the session.
    * Otherwise ($\ell_i = \perp$),
      · Try every $k_\ell = H_3(K_\ell)$ on H_3^{list} as MK to unmask Z to recover (r_C, C) and test $Y = r_C H_1(C)$. If the equation holds, store (r_C, C) into an initially empty list $\mathcal{L}$.
      · If $\mathcal{L}$ is empty, reject the message. (**Rejection 1**)

  - Otherwise, for any tuple (r_C, C) in $\mathcal{L}$, compute $K = \hat{e}(\psi(Y), sbP_2)^{r^t_{i,*}+z} = \hat{e}(\psi(r_C H_1(C)), sbP_2)^{r^t_{i,*}+z} = \hat{e}(r_C \ell_j \psi(sP_2), bP_2)^{(r^t_{i,*}+z)}$, where ℓ_j is from H_1^{list} for identifier C, and $z = H_2(Y, r^t_{i,*}Q_i)$. Note that K is the correct value that party i should compute.
  - Use $MK = H_3(K)$ to unmask Z again to recover $(r^*, \mathtt{ID}^*)$ and test if equation $r^* H_1(\mathtt{ID}^*) = Y$ holds.
  - If the equation holds, set the partner of the oracle as $\mathtt{ID}^*$, $K^t_{i,\mathtt{ID}^*} = K$, and $SK^t_{i,\mathtt{ID}^*} = H_4(\mathtt{ID}^*, i, Y, r^t_{i,\mathtt{ID}^*}Q_i, Z, K^t_{i,\mathtt{ID}^*})$.
  - Otherwise, reject the message. (**Rejection 2**)

- **Send**$^C(\Pi^t_{i,*}, \mathbf{M})$: (Message M is in the form of (j, X)). $\mathcal{B}$ proceeds in the following way:
  - Set the partner of the oracle as j.
  - If $t = J$ and $\ell_j \neq \perp$ which is found from H_1^{list} with identifier j, $\mathcal{B}$ aborts the game (**Event 2**).
  - Otherwise, if $t = J$, $\mathcal{B}$ randomly chooses $u \in \mathbb{Z}_p$ and checks if (uaP_2, X) has been queried on H_2, until one such pair is not found on H_2^{list}. Such u can always be found, because $\mathcal{A}$ is a PPT algorithm of the security parameter k. All the instructions executed before this point belong to the algorithm $\mathcal{B}_1$ of $\mathcal{B}$. $\mathcal{B}$ dumps the content on all the maintained lists and system parameters to the tape σ, and outputs (X, σ). The interactive $\mathrm{BDH}^{\psi}_{2,2,2}$ challenger returns $h \in_R \mathbb{Z}_p$. After this point, all the instructions belong to algorithm $\mathcal{B}_2$ of $\mathcal{B}$. $\mathcal{B}$ reconstructs all the lists and system setup from σ and immediately inserts (uaP_2, X, h) into H_2^{list}. Now $\mathcal{B}$ continues to respond to $\mathcal{A}$'s queries. $\mathcal{B}$ randomly chooses $MK \in \{0,1\}^w, r \in \mathbb{Z}_p$ and generates $Z = (r \| \mathtt{ID}_i) \oplus MK$, and then responds with (uaP_2, Z). If $\mathcal{A}$ rejects the message (**Event 3**), $\mathcal{B}$ randomly chooses K_ℓ from list H_3^{list} and returns $K_\ell^{1/u}$ to the interactive $\mathrm{BDH}^{\psi}_{2,2,2}$ challenger.
  - Otherwise, $\mathcal{B}$ randomly chooses $r \in \mathbb{Z}_p$ and computes $K^t_{i,j} = \hat{e}(\ell_i \psi(sP_2), X + zQ_j)^r$, where ℓ_i and Q_j are found from H_1^{list} with identifier i and j; $z = H_2(rQ_i, X)$. $\mathcal{B}$ computes $Z = (r \| \mathtt{ID}_i) \oplus H_3(K^t_{i,j})$ and $SK^t_{i,j} = H_4(i, j, rQ_i, X, Z, K^t_{i,j})$ and responds with (rQ_i, Z).

- **Reveal**$(\Pi^t_{i,*})$: The oracle must have accepted and so knows its partner j. Otherwise $\perp$ should be returned. If $i \in \mathtt{ID}^C$ and $t = J$, or $i \in \mathtt{ID}^S$ but $\Pi^t_{i,j}$ has a matching conversation with $\Pi^J_{u,v}$ with $u \in \mathtt{ID}^C$, $\mathcal{B}$ aborts the game (**Event 4**). Otherwise, $\mathcal{B}$ returns $SK^t_{i,j}$.

- **Test**$(\Pi^t_{i,*})$: The oracle should be *fresh*, so must have accepted and knows its partner j. If $i \notin \mathtt{ID}^C$ or $t \neq J$, or ($i \in \mathtt{ID}^C$ and $t = J$ but) there is an oracle $\Pi^w_{j,i}$ with the matching conversation to $\Pi^t_{i,j}$ has been revealed, $\mathcal{B}$ aborts the game (**Event 5**). Otherwise, $\mathcal{B}$ randomly chooses a number $\zeta \in \{0,1\}^n$ and gives it to $\mathcal{A}$ as the response.

Once $\mathcal{A}$ responds, $\mathcal{B}$ randomly chooses a tuple from H_4^{list} with the value K_ℓ. $\mathcal{B}$ computes $K_\ell^{1/u}$ and returns it to the interactive $\mathrm{BDH}_{2,2,2}$ challenger. Note that if the game does not abort, for the challenge oracle $K_{i,j}^t = \hat{e}(\ell_i \psi(sP_2), X + hbP_2)^{ua/\ell_i}$.

Let **Event 6** be that, in the attack, the adversary $\mathcal{A}$ indeed chose oracle $\Pi_{i,*}^J$ as the challenger oracle whose recovered partner's identifer $\mathtt{ID}_j$ was queried on H_1 as the I-th distinct identifier query. Then following the rules of the game, it's clear that **Event 1, 2, 4, 5** would not happen, and so the game would not abort before $\mathcal{B}$ can answer the interactive $\mathrm{BDH}_{2,2,2}$ challenge. Hence we have

$$\Pr[\overline{(\texttt{Event 1} \vee \texttt{Event 2} \vee \texttt{Event 4} \vee \texttt{Event 5})}] = \Pr[\texttt{Event 6}] \geq \frac{1}{q_1 \cdot q_o}.$$

Claim 5.6.2. *Let **Event 7** be that $H_3(\hat{e}(\psi(sP_2), Y + hbP_2)^{ua})$ was not queried. Then **Event 3** only happened with negligible probability if **Event 7** and **Event 6** happened.*

Proof: When **Event 6** happened, the agreed secret of the oracle should be $\hat{e}(\psi(sP_2), Y + zbP_2)^{ua}$. Because H_3 is a random oracle, $H_3(\hat{e}(\psi(sP_2), Y + zbP_2)^{ua})$ will be a uniform sample from $\{0,1\}^w$, so is the correct Z. If $H_3(\hat{e}(\psi(sP_2), Y + zbP_2)^{ua})$ was not queried, $\mathcal{A}$ who impersonated the server oracle, can only with negligible probability differentiate the correct mask Z from the faked one $(r \| \mathtt{ID}_i) \oplus MK$ where MK is uniformly sampled from $\{0,1\}^w$.

Claim 5.6.3. *If $MK = H_2(K)$ is not queried, then a message $(Y, (r_C \| C) \oplus MK)$ will be accepted by the server in the real world with only negligible probability.*

Proof: The message will be accepted by the server, only if the recovered message $r_C \| C$ from Z meets the equation $Y = r_C H_1(C)$. Define a relation

$$R = \{(r, \mathtt{ID}, Y) \mid rH_1(\mathtt{ID}) = Y, r \in \mathbb{Z}_p, H_1 \text{ is a hash function}, H_1(\mathtt{ID}) \in \mathbb{G}_2\}.$$

If MK is not queried, the situation can be described by the following game.

$(Y, Z) \leftarrow B^{H_1}(1^k, R)$;
$MK \leftarrow \{0,1\}^w$;
$r_C \| C = Z \oplus MK$;
If $(r_C, C, Y) \in R$, accept, else reject.

If $H_1(C)$ has not been queried, then the equation holds with negligible probability because H_1 is modeled as a random oracle. Suppose, in the attack, q_C distinct client identifiers were queried on H_1. Then for a chosen Y there are at most q_C pairs could possibly meet the equation (i.e. $\mid R \mid = q_C$ in the real attack). Given a mask Z, when MK is a uniformly sampled from $\{0,1\}^w$ independently, the unmask result $r_C \| C$ should have an even distribution among 2^w possibilities. Then the probability that the unmask result is one of q_C pairs (i.e. $(r_C, C, Y) \in R$ where $r_C \| C = Z \oplus MK$), is $\frac{q_C}{2^w}$. Note that in the game both q_C and w are polynomials of security parameter k. Hence the probability is negligible.

Claim 5.6.4. *If $\mathcal{B}$ did not terminate the game, $\mathcal{A}$ cannot notice inconsistence between the simulation and the real world with non-negligible probability.*

Proof: It is clear that if $\mathcal{B}$ did not terminate the game, then the responses to queries including H_1, H_2, H_3, H_4, Corrupt, Send^C, Reveal and Test, are consistent with the ones in the real world. Now let us take a close look at the response to query **Send**$^S(\Pi^t_{i,*}$, **M**). Except for the two rejections, other parts of $\mathcal{B}$'s behavior honestly follow the protocol. **Rejection 1** only happens when $\mathcal{A}$ did not query $MK = H_3(\hat{e}(X, sbP)^{r+z})$. From Claim 5.6.3, in the real world, the server accepts the message with only negligible probability. **Rejection 2** happens if the client message component Z is not valid. This could happen in the attack by a naughty adversary which generates $X = r_C H(C)$, but uses $MK = H_3(T)$ with some other $T \neq K$ to mask $r_C \| C$. If the server accepts the message, the adversary will be quite assured that it is in a simulation, instead of being in the real world.

Following the similar argument as in Claim 5.4.2, we have the following claim.

Claim 5.6.5. *Let **Event 8** be that $K^t_{i,j} = \hat{e}(\psi(sP_2), X + hbP_2)^{ua}$ was not queried on H_4. Then* $\Pr[\overline{\textit{Event 8}} \mid \overline{\textit{Event 3}}] \geq \epsilon(k)$.

Let **Event 9** be the event that $\mathcal{B}$ finds the correct $\hat{e}(\psi(sP_2), X + hbP_2)^{ua}$ on the list H_3^{list} and Let **Event 10** be the event that $\mathcal{B}$ finds the correct $\hat{e}(\psi(sP_2), X + hbP_2)^{ua}$ on the list H_4^{list}.

Overall, we have

$$\begin{aligned}
\Pr[\mathcal{B} \text{ wins}] &= \Pr[\text{Event 6} \wedge \overline{\text{Event 7}} \wedge \text{Event 9} \mid \text{Event 3}] \Pr[\text{Event 3}] \\
&\quad + \Pr[\text{Event 6} \wedge \overline{\text{Event 8}} \wedge \text{Event 10} \mid \overline{\text{Event 3}}] \Pr[\overline{\text{Event 3}}] \\
&\geq \frac{1}{q_1 \cdot q_C \cdot q_3} \Pr[\text{Event 3}] + \frac{1}{q_1 \cdot q_C \cdot q_4} \cdot \epsilon(k) \Pr[\overline{\text{Event 3}}] \\
&\geq \frac{1}{q_1 \cdot q_C} \cdot \frac{1}{\max\{q_3, q_4\}} \cdot \epsilon(k).
\end{aligned}$$

Combining Theorem 5.6.1, the lemma follows. □

(2) *Client authentication.* Now we consider the known-(server)-key attack that the adversary tries to impersonate a client to a server whose private key is known to the adversary or the adversary is benign, i.e. the chosen fresh oracle $\Pi^t_{i,j}$ is with $i \in \mathtt{ID}^S$ and $j \in \mathtt{ID}^C$ and i is corrupted.

Lemma 5.6.3. *The protocol is secure against the known-server-key attack, provided the $GBDH_{2,2,2}$ assumption is sound and the hash functions are modeled as random oracles. Specifically, suppose there is a known-server-key PPT adversary $\mathcal{A}$ against the protocol with non-negligible probability $\epsilon(k)$ and in the attack q_1* client *identifiers are queried on H_1 and q_S* server *oracles are created. Then there exists a PPT algorithm $\mathcal{B}$ to solve the $GBDH_{2,2,2}$ problem with advantage*

$$Adv_{\mathcal{B}}^{GBDH_{2,2,2}}(k) \geq \frac{\epsilon(k)}{q_1 \cdot q_S}.$$

Proof: Given a $\text{GBDH}_{2,2,2}$ problem instance (sP_2, aP_2, bP_2) and a $\text{DBDH}_{2,2,2}$ oracle $\mathcal{O}_{DBDH}$ which given (xP_2, yP_2, zP_2, T) returns 1 if $\hat{e}(P_1, P_2)^{xyz} = T$; otherwise returns 0, we construct an algorithm $\mathcal{B}$ using the adversary $\mathcal{A}$ against the protocol to solve the $\text{GBDH}_{2,2,2}$ problem.

$\mathcal{B}$ simulates the system setup to adversary $\mathcal{A}$ as follows. The system parameters are set as $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, sP_2, H_1, H_2, H_3, H_4)$, i.e. the master secret key is s, which $\mathcal{B}$ does not known; H_1, H_2, H_3, H_4 are random oracles controlled by $\mathcal{B}$.

As in Theorem 5.6.2, we use two types of **Send** query. While, this time we abuse the notation of $\Pi^t_{i,*}$ used in $\mathbf{Send}^S$. We use $\Pi^t_{i,*}$ to denote the t-th server oracle among all the server oracles created in the game.

Algorithm $\mathcal{B}$ randomly chooses $1 \leq I \leq q_1$ and $1 \leq J \leq q_S$ and starts simulating the real world where the adversary $\mathcal{A}$ launches the attack.

- $H_1(\texttt{ID}_i)$: $\mathcal{B}$ maintains an initially empty list H_1^{list} with entries of the form $(\texttt{ID}_i, Q_i, \ell_i)$. $\mathcal{B}$ responds to the query in the following way.
  - If $\texttt{ID}_i$ already appears on H_1^{list} in a tuple $(\texttt{ID}_i, Q_i, \ell_i)$, then $\mathcal{B}$ responds with $H_1(\texttt{ID}_i) = Q_i$.
  - If $\texttt{ID}_i$ is the I-th unique *client* identifier query, then $\mathcal{B}$ inserts $(\texttt{ID}_i, aP_2, \perp)$ into the list and returns aP.
  - Otherwise, $\mathcal{B}$ randomly chooses $\ell_i \in \mathbb{Z}_p$, inserts $(\texttt{ID}_i, \ell_i P_2, \ell_i)$ into the list and returns $\ell_i P_2$.
- $H_2(X_i, Y_i)$: $\mathcal{B}$ maintains an initially empty list H_2^{list} with entries of the form (X_i, Y_i, z_i) indexed by (X_i, Y_i). $\mathcal{B}$ responds to the query in the following way.
  - If a tuple indexed by (X_i, Y_i) is on the list, then $\mathcal{B}$ responds with z_i.
  - Otherwise, $\mathcal{B}$ randomly chooses $z_i \in \mathbb{Z}_p$, inserts (X_i, Y_i, z_i) into the list and returns z_i.
- $H_3(K_\ell)$: $\mathcal{B}$ maintains an initially empty list H_3^{list} with entries of the form (K_i, k_i) indexed by K_i. $\mathcal{B}$ responds to the query in the following way.
  - If a tuple indexed by K_ℓ is on the list, then $\mathcal{B}$ responds with the corresponding k_i.
  - Otherwise, for every tuple $(\Pi^t_{i,j}, u^t, z^t, \ell^t, X^t, Z^t, k^t)$ on list $\mathcal{L}$ which is maintained in the $\mathbf{Send}^C$ routine,
    - Compute $K = \frac{K_\ell^{1/u^t}}{\hat{e}(\psi(aP_2), z^t \ell^t sP_2)}$.
    - Access oracle $\mathcal{O}_{DBDH}(sP_2, aP_2, X^t, K)$.
    - If $\mathcal{O}_{DBDH}$ returns 1, $\mathcal{B}$ removes the entry from $\mathcal{L}$ and sets $K^t_{i,j} = K_\ell$ and $SK^t_{i,j} = H_4(i, j, u^t aP_2, X^t, Z^t, K^t_{i,j})$. Then $\mathcal{B}$ inserts (K_ℓ, k_t) into list H_3^{list} and responds with k_t.

* Otherwise, continue.

– $\mathcal{B}$ randomly chooses $k_\ell \in \{0,1\}^w$, inserts (K_ℓ, k_ℓ) into the list and returns k_ℓ.

- $H_4(\mathtt{ID}_1, \mathtt{ID}_2, Y_1, X_2, Z_1, K_v)$: $\mathcal{B}$ maintains an initially empty list H_4^{list} with entries of the form $(\mathtt{ID}_i, \mathtt{ID}_j, Y_i, X_j, Z_i, K_i, \zeta_i)$ indexed by $(\mathtt{ID}_i, \mathtt{ID}_j, Y_i, X_j, Z_i, K_i)$. $\mathcal{B}$ responds to the query in the following way.

  – If a tuple indexed by $(\mathtt{ID}_1, \mathtt{ID}_2, Y_1, X_2, Z_1, K_v)$ is on the list, then $\mathcal{B}$ responds with the corresponding ζ_i.

  – Otherwise, for every tuple $(\Pi_{i,j}^t, u^t, z^t, \ell^t, X^t, Z^t, k^t)$ on list $\mathcal{L}$,

    * Compute $K = \frac{K_v^{1/u^t}}{\hat{e}(\psi(aP_2), z^t x^t s P_2)}$.
    * Access oracle $\mathcal{O}_{DBDH}(sP_2, aP_2, X^t, K)$.
    * If $\mathcal{O}_{DBDH}$ returns 1, $\mathcal{B}$ removes the entry from $\mathcal{L}$ and sets $K_{i,j}^t = K_v$ and $SK_{i,j}^t = H_4(i, j, u^t a P_2, X^t, Z^t, K_{i,j}^t)$. $\mathcal{B}$ inserts (K_v, k^t) into list H_3^{list}. If $\mathtt{ID}_1 = i, \mathtt{ID}_2 = j, Y_1 = u^t a P_2, X_2 = X^t$, and $Z_1 = Z^t$, then $\mathcal{B}$ responds with $SK_{i,j}^t$.
    * Otherwise, continue.

  – $\mathcal{B}$ randomly chooses $\zeta_i \in \{0,1\}^n$ and inserts $(\mathtt{ID}_1, \mathtt{ID}_2, Y_1, X_2, Z_1, K_v, \zeta_i)$ into the list and returns ζ_i.

- **Corrupt**$(\mathtt{ID}_i)$: $\mathcal{B}$ looks through list H_1^{list}. If $\mathtt{ID}_i$ is not on the list, $\mathcal{B}$ queries $H_1(\mathtt{ID}_i)$. $\mathcal{B}$ checks the value of ℓ_i: if $\ell_i \neq \perp$, then $\mathcal{B}$ responds with $\ell_i s P_2$; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).

- **Send**$^S(\Pi_{i,*}^t$, **M**): $\mathcal{B}$ maintains a list Ω for every (client or server) oracle of the form $(\Pi_{i,j}^t, tran_{i,j}^t, r_{i,j}^t, K_{i,j}^t, SK_{i,j}^t)$ where $tran_{i,j}^t$ is the transcript of the oracle so far; $r_{i,j}^t$ is the random integer used by the oracle to generate message, and $K_{i,j}^t$ and $SK_{i,j}^t$ are set $\perp$ initially. $\mathcal{B}$ proceeds in the following way:

  – If $t = J$,

    * If $M = \lambda$, $\mathcal{B}$ responds with (i, bP_2) and sets $r_{i,*}^t = \perp$.
    * Otherwise (i.e. $M = (Y, Z)$), $\mathcal{B}$ tries every k_i on H_3^{list} and k^t on $\mathcal{L}$ as MK to umask Z to recover (r_C, C) and tests $Y = r_C H_1(C)$. If no pair meets the equation, or if for all the recovered messages (r_C, C) unmasked using some k_w's, ℓ_C on H_1^{list} corresponding to C is not $\perp$, $\mathcal{B}$ aborts the game (**Event 2**). Otherwise, there is at least a $k_w = H_3(K_w)$ unmasking a valid C whose corresponding ℓ_C on H_1^{list} is $\perp$, i.e. $H_1(C) = aP_2$.

  – Otherwise,

    * If $M = \lambda$, $\mathcal{B}$ randomly chooses $r_{i,*}^t \in \mathbb{Z}_p$ and responds with $(i, r_{i,*}^t Q_i)$ where Q_i is found from H_1^{list} with identifier i;

* Otherwise ($M = (Y, Z)$), $\mathcal{B}$ computes $K = \hat{e}(\psi(Y), \ell_i sP)^{r^t_{i,*}+z}$, where ℓ_i is from H_1^{list} for identifier i; $r^t_{i,*}$ is from list Ω for oracle $\Pi^t_{i,*}$ and $z = H_2(Y, r^t_{i,*}Q_i)$. $\mathcal{B}$ unmasks Z to recover (r_C, C) using $MK = H_3(K)$ and tests $Y = r_C H_1(C)$. If the equation does not hold, $\mathcal{B}$ rejects the message; otherwise, $\mathcal{B}$ sets the partner identifier as C, $K^t_{i,C} = K$ and computes $SK^t_{i,C} = H_4(C, i, Y, r^t_{i,C}Q_i, Z, K^t_{i,C})$.

- **Send**$^C(\Pi^t_{i,*}, \mathbf{M})$: (Message M is in the form of (j, X)). $\mathcal{B}$ maintains an initially empty list $\mathcal{L}$ with entries of the form $(\Pi^t_{i,j}, u^t, z^t, \ell^t, X^t, Z^t, k^t)$. $\mathcal{B}$ proceeds in the following way:

  - Set the partner of the oracle as j and find ℓ_j from H_1^{list} with identifier j.
  - If $\ell_i = \perp$ from H_1^{list} with identifier i, $\mathcal{B}$ proceeds in the following way.
    * Randomly chooses $u \in \mathbb{Z}_p$ and computes $z = H_2(uaP_2, X)$. Hence the agreed secret of this oracle should be

$$\begin{aligned} K^t_{i,j} &= \hat{e}(\psi(saP_2), X + z\ell_j P_2)^u \\ &= (\hat{e}(\psi(saP_2), X) \cdot \hat{e}(\psi(aP_2), z\ell_j sP_2))^u \end{aligned}$$

    * For every (K_ℓ, k_ℓ) on H_3^{list},
      · Compute $K = \frac{K_\ell^{1/u}}{\hat{e}(\psi(aP_2), z\ell_j sP_2)}$.
      · Access oracle $\mathcal{O}_{DBDH}(sP_2, aP_2, X_2, K)$.
      · If $\mathcal{O}_{DBDH}$ returns 1, $\mathcal{B}$ sets $K^t_{i,j} = K_\ell$ and computes $Z = (u\|i) \oplus k_\ell$ and $SK^t_{i,j} = H_4(i, j, uaP_2, X, Z, K^t_{i,j})$. Then $\mathcal{B}$ responds with (uaP_2, Z).
      · Otherwise, continue.
    * For every $(\cdots, K_v, \zeta_v)$ on H_4^{list},
      · Compute $K = \frac{K_v^{1/u}}{\hat{e}(\psi(aP_2), z\ell_j sP_2)}$.
      · Access oracle $\mathcal{O}_{DBDH}(sP_2, aP_2, X, K)$.
      · If $\mathcal{O}_{DBDH}$ returns 1, $\mathcal{B}$ sets $K^t_{i,j} = K_v$. Then $\mathcal{B}$ uses $k = H_3(K^t_{i,j})$ as MK to mask $u\|i$ and responds with $(uaP_2, (u\|i) \oplus MK)$.
      · Otherwise, continue.
    * It is highly unlikely that the above two searches can find proper response, because $K^t_{i,j}$ is depending on z and u which are just generated randomly. So, the searches above can be omitted. Instead, $\mathcal{B}$ directly randomly chooses $k \in \{0,1\}^w$, computes $Z = (u\|i) \oplus k$ and stores the tuple $(\Pi^t_{i,j}, u, z, \ell_j, X, Z, k)$ into list $\mathcal{L}$. $\mathcal{B}$ responds with (uaP_2, Z).
  - Otherwise, $\mathcal{B}$ randomly chooses $r \in \mathbb{Z}_p$ and computes $K^t_{i,j} = \hat{e}(\ell_i \psi(sP_2), X + zQ_j)^r$, where ℓ_i and Q_j are found from H_1^{list} with identifier i and j; $z = H_2(rQ_i, X)$. $\mathcal{B}$ computes $Z = (r\|i) \oplus H_3(K^t_{i,j})$ $SK^t_{i,j} = H_4(i, j, rQ_i, X, Z, K^t_{i,j})$ and responds with (rQ_i, Z).

- **Reveal**($\Pi^t_{i,*}$): The oracle must have accepted and so knows its partner j. Otherwise $\perp$ should be returned. If $i \in \mathtt{ID}^S$ and $t = J$, or $i \in \mathtt{ID}^C$ but has a matching conversation to $\Pi^J_{u,v}$ with $u \in \mathtt{ID}^S$, $\mathcal{B}$ aborts the game (**Event 3**). Otherwise, $\mathcal{B}$ returns $SK^t_{i,j}$.
- **Test**($\Pi^t_{i,*}$): The oracle should be *fresh*, so must have accepted and knows its partner j. If $i \notin \mathtt{ID}^S$ or $t \neq J$, or ($i \in \mathtt{ID}^S$ and $t = J$ but) there is an oracle $\Pi^w_{j,i}$ with the matching conversation to $\Pi^t_{i,j}$ has been revealed, then $\mathcal{B}$ aborts the game (**Event 4**). Otherwise, $\mathcal{B}$ randomly chooses a number $\zeta \in \{0,1\}^n$ and gives it to $\mathcal{A}$ as the response.

Once $\mathcal{A}$ responds, $\mathcal{B}$ proceeds in the following way. For all the K_v on H_4^{list},

- Compute $K = (\frac{K_v}{\hat{e}(r_C\psi(aP_2),\ell_i sP_2)^z})^{1/r_C}$ where $z = H_2(Y, bP_2)$ and (Y, Z) is the incoming message to $\Pi^J_{i,*}$ and r_C and C are recovered in the $\text{Send}^S(\Pi^J_{i,*}, M)$.
- Access oracle $\mathcal{O}_{DBDH}(sP_2, aP_2, bP_2, K)$.
- If $\mathcal{O}_{DBDH}$ returns 1, return K as the answer to the GBDH problem. Note that the agreed secret of tested oracle should be

$$\begin{aligned} K^J_{i,*} &= \hat{e}(\psi(Y), \ell_i sP_2)^{r^J_{i,*}+z} \\ &= \hat{e}(r_C\psi(aP_2), \ell_i sP_2)^{r^J_{i,*}+z} \\ &= \hat{e}(\psi(aP_2), bsP_2)^{r_C} \cdot \hat{e}(r_C\psi(aP_2), \ell_i sP_2)^z \text{ since } r^J_{i,*} = b/\ell_i. \end{aligned}$$

- If for all K_v, the test fails and $\mathcal{B}$ fails the game.

Claim 5.6.6. *If $\mathcal{B}$ did not abort the game, $\mathcal{A}$ could not find inconsistency between the simulation and the real world with non-negligible probability.*

Proof: The response to queries are indistinguishable from the one in the real world. In particular, to respond to queries on H_3 and H_4, $\mathcal{B}$ significantly uses the access to $\mathcal{O}_{DBDH}$ and the programmability of a random oracle to make sure that the response is consistent with the one in Send^C.

By the rule of the game, the message (Y, Z) to the chosen test oracle has to be accepted. Hence the recovered value $r_C \| C$ from Z has to meet the equation $r_C H_1(C) = Z$. If C is not queried on H_1, which is a random oracle, the message passes the check with only negligible probability. So, the identifier of $\Pi^J_{i,*}$'s partner must have been queried on H_1 in the game.

Let **Event 5** be that, in the attack, the adversary $\mathcal{A}$ indeed chose oracle $\Pi^J_{i,*}$ as the challenger oracle whose recovered partner's identifer $\mathtt{ID}_j$ was queried on H_1 as the I-th distinct identifier query. Then following the rules of the game, it's clear that **Event 1, 3, 4** would not happen. And

$$\Pr[\text{Event } 5] \geq \frac{1}{q_1 \cdot q_S}.$$

Claim 5.6.7. ***Event 2*** *happened with negligible probability if* ***Event 5*** *happened*

Proof: There are two possibilities that **Event 2** could happen. 1) For the J-th server oracle, the partner oracle's identifier $\mathtt{ID}^*$ was not queried on H_1 as the I-th one. While. this could not happen if **Event 5** happened. 2) $MK = H_3(\hat{e}(\psi(X), sP_2)^{b+z})$ is not queried (we note that $\mathcal{B}$ simulates some queries on H_3 in responding to Send^C query, though it does not know the query value), but the message (Y, Z) passes the check $Y = r_C H_1(C)$ where $r_C \| C = Z \oplus MK$ to be accepted by the server. From Claim 5.6.3, this will only happen with negligible probability.

Following the similar argument as in Claim 5.4.2, we have the following claim.

Claim 5.6.8. *Let **Event 6** be that $K = \hat{e}(\psi(Y), \ell_i sP_2)^{r_{i,*}^J + z}$ was not queried on H_4. Then* $\Pr[\overline{\textit{Event 6}}] \geq \epsilon(k)$.

Overall, we have

$$\Pr[\mathcal{B} \text{ wins}] = \Pr[\texttt{Event 5} \wedge \overline{\texttt{Event 2}} \wedge \overline{\texttt{Event 6}}] \geq \frac{1}{q_1 \cdot q_S} \cdot \epsilon(k).$$

This completes the proof. □

Lemma 5.6.2 and 5.6.3 show that the protocol possesses the following security properties: mutual key authentication, known session key security, key-compromise impersonation resilience and unknown key-share resilience. Based on these results and the fact that the GBDH assumption implies the corresponding BDH assumption, it is straightforward that the following theorem holds.

Theorem 5.6.4. *The protocol is a secure AK, provided the $GBDH_{2,2,2}$ assumption is sound and the hash functions are modeled as random oracles.*

Now let's look at the the forward secrecy. Note that a protocol satisfies Definition 5.2.2 does not necessary achieve the forward secrecy. The protocol in Figure 5.6 achieves PFS.

Theorem 5.6.5. *The protocol has PFS regarding Definition 5.2.3, provided that the $BDH_{2,2,2}$ assumption is sound and the hash functions are modeled as random oracles. Specifically, suppose there is a PPT adversary $\mathcal{A}$ against the protocol's PFS with non-negligible advantage $\epsilon(k)$ and in the attack adversary $\mathcal{A}$ creates q_C client oracles and q_S server oracles and queries q_3 and q_4 times on H_3 and H_4 respectively. Then there exists a PPT algorithm $\mathcal{B}$ to solve the $BDH_{2,2,2}$ problem with advantage*

$$Adv_{\mathcal{B}}^{BDH_{2,2,2}}(k) \geq \frac{\epsilon(k)}{q_C \cdot q_S \cdot (q_3 + q_4)}.$$

Proof (sketch): If the adversary $\mathcal{A}$ can win the game by choosing some session between party I and J, given (aP_2, bP_2, sP_2) we construct an algorithm $\mathcal{B}$ for the BDH problem as follows. $\mathcal{B}$ sets s as the master secret key which it does not know, i.e. $R = sP_2$ in the system public parameters. Without loosing generality we assume that $Q_I = xP_2$ is the client's public key and $Q_J = yP_2$ is the server's public key for some randomly chosen $x, y \in \mathbb{Z}_p$ by $\mathcal{B}$. $\mathcal{B}$ sets $D_I = sxP_2$ and $D_J = syP_2$. In the attacking session, the algorithm sets $r_I Q_I = aP_2$ and $r_J Q_J = bP_2$, i.e. $r_I = a/x$ and $r_J = b/y$. Note that according to the rules of the game, the adversary should not tamper with the messages in the challenge session. Then the algorithm computes $T = \hat{e}(P_1, P_2)^{sab} = \hat{e}(\psi(Q_I), Q_J)^{sr_I r_J} = \frac{K}{\hat{e}(\psi(r_I Q_I), D_J)^h}$, where $h = H_2(aP_2, bP_2)$ and K is the established secret of the session which is computed by the adversary subroutine. Note that $\mathcal{A}$ has to compute K to win the game with non-negligible advantage because H_4 is a random oracle. While, in the proof, there is a pitfall: in the challenge session, $\mathcal{B}$ has to generate a client message (aP_2, Z) in which it does not know both a/x (the random flips used by the client I in the challenge session) and the mask key MK. Fortunately, $\mathcal{B}$ only needs to generate a message which cannot be differentiated by the adversary from the valid one. $\mathcal{B}$ can randomly sample $MK \in \{0,1\}^w$ to mask a dummy value for a/x. As H_3 is a random oracle, if $\mathcal{A}$ notices the difference between the faked Z and the valid one, it must have queried K on H_3, then $\mathcal{B}$ can solve the BDH problem. These tricks have been played in the proof of Lemma 5.6.2. □

User identity privacy is achieved for the following reasons. (1) We note that r_A is randomly sampled from $\mathbb{Z}_p$, hence $r_A Q_A$ is evenly distributed in message space $\mathbb{G}_2$ which could be sampled by any other client with the same distribution. Therefore $r_A Q_A$ for client $\mathcal{B}$ is indistinguishable from $r_X Q_X$ for any other client. (2) The client's identity is hidden by the mask operation in the second flow. The used mask key is the output from a hash function on the agreed secret so that it is evenly distributed in the key space if we assume the hash function can be modeled as a random oracle. Hence, the masked result is also

evenly distributed in the message space and indistinguishable from random.

It is clear that, if the adversary knows the master secret key s, then it can compute K, so that the scheme does not achieve the master key forward secrecy. However, this property may not necessarily be a defect since it provides a possible way for legal interception which is also important in mobile communications.

A few points of the protocol appear worth mentioning. In Section 5.4.4, we have discussed the importance of testing the group membership of exchanged messages. The protocol integrates the test of $r_A Q_A$ ($r_A \psi(Q_A)$ resp.)'s membership by checking $r_C H_1(C) \stackrel{?}{=} r_A Q_A$ ($r_C \psi(H_1(C)) \stackrel{?}{=} r_A \psi(Q_A)$ resp.). If the test of $r_B Q_B \in \mathbb{G}_2$ is too costly, one may choose to ignore it with certain risk. The protocol provides some protection on this "lazy" implementation. Before $r_B Q_B$ is used, it has been masked by a random point hQ_A. One choosing $r_B Q_B$ cannot decide the representation of $r_B Q_B + hQ_A$. Moreover, for the Tate pairing the second input can be any point on the curve, which does not decide the order of the pairing result. Interestingly, this has been echoed in the security analysis in Section 5.6.3. When dealing with an adversary who chooses $r_A Q_A$, the security reduction is based on the gap BDH assumption, which requires that messages lie in the right group for the decisional oracle to work. Hence the check $r_C H_1(C) \stackrel{?}{=} r_A Q_A$ is crucial both for the proof to go through and for the security of the protocol. Note that $r_B Q_B$ is chosen properly in this case by a legitimate party. When dealing with an adversary who chooses $r_B Q_B$, the security reduction is based on the BDH assumption, which does not pose such requirement explicitly. However, the attacks in [123, 129] on HMQV [106] and the attack in Section 5.4.4 show that proofs should be carefully interpreted, which appears worth further study. A cautious measure is to fulfil the assumption that messages are with the right representation with some form of validation.

5.6.4 Efficiency Discussion and Comparison

All the existing protocols discussed in Section 5.3 do not provide identity privacy. However, as IKE [95], we can tweak some protocols such as the SCK and SYL, which use Diffie-Hellman tokens, to provide identity privacy against certain adversaries. IB-KAPs require peer party's identifer to compute the shared secret, which will be used for entity authentication. Hence we have to slightly modify IKE to adapt to this requirement. Figure 5.7 presents a simplified identity-based IKE from pairing, which may be of independent interest.

$(Initiator)A$		$B(Responder)$
$Q_A = H_1(A)$		$Q_B = H_1(B)$
$r_A \in_R \mathbb{Z}_p$	$\xrightarrow{r_A P_2}$	
		$r_B \in_R \mathbb{Z}_p$;
		$K_1 = H_2(r_A P_2, r_B P_2, r_A r_B P_2)$;
	$\xleftarrow{r_B P_2, Z_1}$	$Z_1 = ENC_{K_1}(B)$
$K_1 = H_2(r_A P_2, r_B P_2, r_A r_B P_2)$;		
$B = ENC_{K_1}^{-1}(Z_1)$;		
$K_2 = K_{SCK}$ or K_{SYL};		
$AUTH_A = H_3(K_2, A, B, r_A P_2, r_B P_2, Z_1)$;		
$Z_2 = ENC_{K_1}(A \Vert AUTH_A)$	$\xrightarrow{Z_2}$	
		$A \Vert AUTH_A = ENC_{K_1}^{-1}(Z_2)$;
		$K_2 = K_{SCK}$ or K_{SYL};
		$AUTH_A \stackrel{?}{=} H_3(K_2, A, B, r_A P_2, r_B P_2, Z_1)$
		$AUTH_B = H_3(K_2, B, A, r_A P, r_B P_2, Z_1, Z_2)$;
	$\xleftarrow{Z_3}$	$Z_3 = ENC_{K_1}(AUTH_B)$
$AUTH_B = ENC_{K_1}^{-1}(Z_3)$;		
$AUTH_B \stackrel{?}{=} H_3(K_2, B, A, r_A P, r_B P_2, Z_1, Z_2)$		
$SK = H_4(K_2)$		$SK = H_4(K_2)$

1. H_i is a cryptographic hash function for $i = 1, \ldots, 4$.
2. $ENC_K(M)$ is a symmetric key encryption scheme encrypting message M with secret key K and $ENC_K^{-1}(C)$ is the decryption operation with the ciphertext C and secret key K.
3. K_{SCK} (resp. K_{SYL}) is the established secret by the SCK (resp. SYL) scheme including the DH value.

Figure 5.7: Simplified Identity-based IKE

IKE provides identity privacy for both the initiator and the responder against passive adversaries, but does not protect the identity privacy of the initiator against an active

adversary who impersonates the responder. The identity-based IKE protocol in Figure 5.7 does not guarantee identity privacy for either party against active adversaries. One may use the IKE main mode authenticated with identity-based signatures [93, 45, 25]. The protocol then involves the signature signing and verification process which can be costly. Even the efficient BLMQ-IBS [25] requires two exponentiations in $\mathbb{G}_t$, two multiplications in $\mathbb{G}_1$ and one pairing. The CC-IBS [45] and Hess-IBS [93] are slower than BLMQ-IBS in general implementation.

The comparative efficiency and security properties of the presented IB-KAPs with identity privacy are summarised in Table 5.2. For simplicity, we only consider the Type-1 pairings. From the results we can see that our proposal in Figure 5.6 ranks among the fastest schemes.

Table 5.2: Efficiency and Security Comparison

Schemes	Key Const.	KSK	p-FS	KCI	UKS	IP	Computation
The Proposal	SOK	✓	✓	✓	✓	✓(1)	$1P + 2M + 1E$
IKE with SCK	SOK	✓	✓	✓	✓	Passive(2)	$2P + 3M$
IKE with SYL	SOK	✓	✓	✓	✓	Passive(2)	$1P + 3M$
IKE with Hess-IBS	SOK	✓	✓	✓	✓	✓(3)	$2P + 4M + 2E$
IKE with CC-IBS	SOK	✓	✓	✓	✓	✓(3)	$2P + 5M$
IKE with BLMQ-IBS	SK	✓	✓	✓	✓	✓(3)	$1P + 4M + 2E$

1. Identity privacy is only preserved for the client (the responder).
2. Identity privacy is only achievable against passive adversaries.
3. Identity privacy is achievable for both parties against passive adversaries, but only achievable for the responder against active adversaries.

The IKE protocol in Figure 5.7 requires four message flows. For the protocol authenticated with signatures, three message flows are required at least to establish a shared secret like the Station-to-Station protocol [127]: The responder can include the encryption of its identity and signature on the Diffie-Hellman tokens in the second messages along with $r_B P_2$. The initiator responds with the encryption of its identity and signature on the exchanged messages so far in the third message. The proposed protocol with two message flows in

Figure 5.6 is more round efficient to establish a shared secret than other protocols.

5.7 Conclusion

In this chapter, we have investigated the identity-based two-party authenticated key agreement protocols from pairings. We formally analysed several practically efficient this type of key agreements, including the enhanced Smart protocol, the enhanced Shim protocol and the McCallugh-Barreto protocols, in the enhanced Bellare-Rogaway key agreement model. We provided proofs for the first two schemes based on the weakest possible assumption (BDH) for this type of protocol by adopting a new technique: the built-in decisional function. This has not been accomplished before and the method can be used to analyse some other protocols with similar structure. For the McCallugh-Barreto protocols, we first pointed out the errors in the existing security analyses and then slightly tweaked the MB-2 protocol and formally proved its security but based on the strong ℓ-GBCAA1 assumption. The formal analyses build strong confidence in the security of these schemes. Then, in considering that the user identity may be sensitive information in many environments, we proposed a special identity-based key agreement, which achieves unilateral user identity privacy, and formally analysed its security. We also demonstrated the importance of verifying that the exchange message possesses certain arithmetic properties in this type of protocol and showed how to process the group membership check for such purpose.

Chapter 6

Conclusions and Open Problems

The (Weil or Tate) pairings on elliptic curves used to be considered to be destructive in ECC. Only recently, after some pioneering works with pairings, such as the Joux's pairing-based tripartite key agreement and the famous Boneh-Franklin IBE, pairings have been quickly become an essential tool to construct many novel cryptographic schemes. Despite some doubts on the new pairing-related complexity assumptions, pairing-based cryptography has been gathering momentum in practice [160].

As other fields in cryptography, pairing-based cryptography research mainly consists of two types of activity: *definitional activities* which identify concepts that were not known before, and *constructive activities* which either demonstrate the plausibility of a cryptographic definition in the standard model or design efficient schemes for practical applications within commonly accepted paradigms, such as the random oracle model. This thesis is devoted to the study of pairing-based cryptography, but has primarily focused on efficient schemes suitable for practical applications. This requires the constructed schemes not only to meet robust security definitions but also to achieve high efficiency.

In this work, three pairing-based cryptographic primitives: IBE, CL-PKE and IB-KAP were studied.

On IBE, using the Sakai-Kasahara identity key construction, we constructed a complete

IBE (SK-IBE) with provable security in the Boneh-Franklin IBE model based on a reasonable complexity assumption ℓ-BCAA1. With the same key construction, two ID-KEMs (SK-KEM1 and SK-KEM2) were proposed. The first bases its security on a gap assumption ℓ-GBCAA1, while the second can be proven to be secure with ℓ-BCAA1. Owing to the used key construction, three schemes achieve the best performance among all the existing schemes of the type. Depending on the chosen pairing parameters, the proposed schemes can be 50% to 300% faster in either encryption or decryption than other schemes. SK-KEM2 has already been adopted in commercial products.

On CL-PKE, we proposed a heuristic approach to constructing efficient CL-PKE schemes, namely, when an IBE shares a similar structure with a PKE, using a hash function to bind the two schemes can result in a CL-PKE. Following this approach, we constructed three CL-PKEs using three types of IBE, which are secure in the enhanced Al-Riyami-Paterson CL-PKE model. Each of the three CL-PKEs has the same ciphertext size as the used IBE and requires only an extra point multiplication than the based IBE. As the used IBEs are the efficient one of each type, the CL-PKEs exhibit high performance.

On IB-KAP, several practically efficient schemes including the enhanced Smart protocol, the enhanced Shim protocol and the McCallugh-Barreto protocols have been analysed in the enhanced Bellare-Rogaway key agreement model. By adopting the built-in decisional function in the reduction, the security of the first two schemes, which have not been formally analysed before, were proven based on the weakest possible assumption for this type of protocol. This is the first time that a reduction based on the BDH assumption has been accomplished for a protocol exhibiting all the common security properties including the key compromise impersonation resilience. For the McCallugh-Barreto protocols, serious errors in the existing security analysis were pointed out first, and then a slightly tweaked MB-2 protocol was formally analysed but based on the stronger ℓ-GBCAA1 assumption. The formal analysis builds strong confidence in the security of these schemes, particularly for

standardisation. The work has been submitted to IEEE P1363.3 as a contribution to the pairing-based cryptography standard. Finally, in considering that the user identity may be sensitive information in many environments, a special IB-KAP was proposed, which achieves unilateral user identity privacy and is efficient in both computation and communication, and its security was formally analysed.

Although in the last few years pairing has been extensively used as a building block to construct many novel cryptographic protocols [21], we believe that its potential in cryptography is far from fully explored. On the topics investigated in this thesis, several problems of theoretical and practical interest remain to be answered.

On the extensibility of SK-IBE/KEM, though the schemes exhibit high efficiency, so far they still lack certain extensions which may be important in certain environments. It remains interesting to extend the SK-IBE/KEM to a hierarchal system and to build a robust system such as a secret sharing scheme for the SK key construction to protect the master secret.

On the applicability of CL-PKE in practice, two questions, namely, how to distribute public keys and how to revoke public keys, have to be answered before CL-PKE becomes a viable cryptographic solution. The problem of constructing CL-PKE secure in the stronger model (Type-I+Type-II$^+$ security) without random oracles still remains open. It is theoretically important to see whether certain constructions admitting proofs in the strong models with random oracles imply a separation between the random oracle model and the standard model. Formal analysis of the approach to CL-PKE proposed in Section 4.4 in a more general framework is also worth study.

On the security analysis of cryptographic schemes, we have seen the proofs have become more and more complicated and error-prone. It is important to investigate potential approaches to making the proofs verifiable. There have been several works of this line such as the game hopping techniques [147, 35] and the computer-aided symbolic proofs with

cryptographic soundness [55, 92]. Certainly more work needs to be done in this area.

Bibliography

[1] M. Abadi. Private authentication. In *Proc. of Privacy Enhancing Technologies 2002*, LNCS 2482, pp. 27–40, 2002.

[2] S. Al-Riyami. Cryptographic schemes based on elliptic curve pairings. PhD thesis, Royal Holloway, University of London, 2004.

[3] M. H. Au, J. Chen, J. K. Liu, Y. Mu, D. S. Wong and G. Yang. Malicious KGC attack in certificateless cryptography. Cryptology ePrint Archive, Report 2006/255, 2006.

[4] N. Attrapadung, B. Chevallier-Mames, J. Furukawa, T. Gomi, G. Hanaoka, H. Imai and R. Zhang. Efficient identity-based encryption with tight security reduction. In *Proc. of CANS06*, LNCS 4301, pp. 19–36, 2005. Also available on Cryptology ePrint Archive, Report 2005/320, 2005.

[5] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *Proc. of Advances in Cryptology - Asiacrypt 2003*, LNCS 2894, pp. 452–473, 2003.

[6] S. S. Al-Riyami and K. G. Paterson. CBE from CL-PKE: a generic construction and efficient schemes. In *Proc. of Public Key Cryptography - PKC 2005*, LNCS 3386, pp. 398–415, 2005.

[7] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptology* 15(2):103-127, 2002.

[8] M. Burrows, M. Abadi and R. Needham. A logic for authentication. DEC Systems Research Center Technical Report 39, 1990.

[9] X. Boyen. The BB_1 identity-based cryptosystem: a standard for encryption and key encapsulation.

http://grouper.ieee.org/groups/1363/IBC/submissions/Boyen-bb1_ieee.pdf, August 2006.

[10] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Proc. of Advances in Cryptology - Eurocrypt 2004*, LNCS 3027, pp. 223–238, 2004.

[11] D. Boneh and X. Boyen. Short signatures without random oracles. In *Proc. of Advances in Cryptology - Eurocrypt 2004*, LNCS 3027, pp. 56–73, 2004.

[12] D. Boneh and X. Boyen. Secure identity-based encryption without random oracles. In *Proc. of Advances in Cryptology - Crypto 2004*, LNCS 3152, pp. 443–459, 2004.

[13] M. Bellare, A. Boldyreva and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Proc. of Advances in Cryptology - Eurocrypt 2004*, LNCS 3027, 2004.

[14] M. Barbosa, L. Chen, Z. Cheng, M. Chimley, A. Dent, P. Farshim, K. Harrison, J. Malone-Lee, N.P. Smart and F. Vercauteren. SK-KEM : an identity-based KEM. Sumission to IEEE P1363.3, 2006.

[15] M. Bellare, R. Canetti and H. Krawczyk. Keying hash functions for message authentication. In *Proc. of Advances in Cryptology - Crypto '96*, LNCS 1109, pp. 1–15, 1996.

[16] M. Bellare, R. Canetti and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proc. of the 30th STOC*, pp. 419–428, ACM Press, 1998.

[17] E. Bresson, O. Chevassut and D. Pointcheval. Provably authenticated group Diffie-Hellman key exchange - the dynamic case. In *Proc. of Advances in Cryptology - Asiacrypt 2001*, LNCS 2248, pp. 290–309, 2001.

[18] E. Bresson, O. Chevassut and D. Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In *Proc. of Advances in Cryptology - Eurocrypt 2002*, LNCS 2332, pp. 321–336, 2002.

[19] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Proc. of Advances in Cryptology - Crypto 2001*, LNCS 2139, pp. 213–229, 2001.

[20] K. Bentahar, P. Farshim, J. Malone-Lee and N. P. Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058, 2005.

[21] P. Barreto. The pairing based crypto lounge. `http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html`.

[22] P. Barreto, S. Galbraith, C. ÓhÉigeartaigh and M. Scott. Efficient pairing computation on supersingular abelian varieties. Cryptology ePrint Archive, Report 2004/375, 2004.

[23] S. Blake-Wilson, D. Johnson and A. Menezes. Key agreement protocols and their security analysis. In *Proc. of Cryptography and Coding*, LNCS 1355, pp. 30–45, 1997.

[24] P. Barreto, H. Kim, B. Lynn and M. Scott. Efficient implementation of pairing-based cryptosystems. *J. Cryptology*, 17(4):321–34, 2004.

[25] P. Barreto, B. Libert, N. McCullagh and J.-J. Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Proc. of Advances in Cryptology - Asiacrypt 2005*, LNCS 3788, pp. 515–532, 2005.

[26] C. Boyd, W. Mao and K. Paterson. Key agreement using statically keyed authenticators. In *Proc. of Applied Cryptography and Network Security: Second International Conference - ACNS*, LNCS 3089, pp. 248–262, 2004.

[27] C. Boyd and A. Mathuria. Key establishment protocols for secure mobile communications: a selective survey. In *Proc. of Australasian Conference on Information Security and Privacy'98*, LNCS 1438, pp. 344–355, 1998.

[28] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Proc. of SAC 2005*, LNCS 3897, pp. 319–331, 2006. Also available on Cryptology ePrint Archive, Report 2005/133.

[29] J. Baek, R. Safavi-Naini and W. Susilo. Certificateless public key encryption without pairing. In *Proc. of the 8th International Conference on Information Security (ISC 2005)*, LNCS 3650, pp.134–148, 2005.

[30] M. Bellare, D. Pointcheval and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proc. of Advances in Cryptology - Eurocrypt 2000*, LNCS 1807, pp. 139–155, 2000.

[31] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of the 1st CCS*, pp. 62–73, 1993.

[32] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proc. of Advances in Cryptology - Crypto '93*, LNCS 773, pp. 232–249, 1993.

[33] M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with RSA. In *Proc. of Advances in Cryptology - Eurocrypt '94*, LNCS 950, pp. 92–111, 1995.

[34] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *Proc. of the 27th STOC*, pp. 57–66, ACM Press, 1995.

[35] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - Eurocrypt 2006*, LNCS 4004, pp. 409–426, 2006.

[36] I. F. Blake, G. Seroussi and N. Smart. Elliptic curves in cryptography. *volume 265 of London Mathematical Society Lecture Notes*, Cambridge University Press, 1999.

[37] I. F. Blake, G. Seroussi and N. Smart. Advances in elliptic curve cryptography. *volume 317 of London Mathematical Society Lecture Notes*, Cambridge University Press, 2005.

[38] S. Blake-Wilson and A. Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. In *Proc. of Security Protocols Workshop '97*, LNCS 1361, pp. 137–158, 1997.

[39] X. Boyen, Q. Mei and B. Waters. Direct chosen ciphertext security from identity-based techniques. In *Proc. of ACM Conference on Computer and Communications Security CCS 2005*, pp. 320–329, ACM Press, 2005.

[40] Z. Cheng. Simple SK-ID-KEM. notes, June, 2005.

[41] J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In *Proc. of Advances in Cryptology - Eurocrypt 2006*, LNCS 4004, pp. 1–11, 2006.

[42] C. Cocks. An identity-based encryption scheme based on quadratic residues. In *Proc. of Cryptography and Coding*, LNCS 2260, pp. 360–363, 2001.

[43] K. Choo, C. Boyd and Y. Hitchcock. On session key construction in provably-secure key establishment protocols: revisiting Chen & Kudla (2003) and McCullagh & Barreto (2005) ID-Based Protocols. Cryptology ePrint Archive, Report 2005/206, 2005.

[44] S. Chow, C. Boyd and J. Nieto. Security-mediated certificateless cryptography. In *proc. of Public Key Cryptography 2006*, LNCS 3958, pp. 508–524.

[45] J. C. Cha and J. H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Proc. of Public Key Cryptography PKC 2003*, LNCS 2567, pp. 18–30, 2003.

[46] Z. Cheng and R. Comley. Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012, 2005.

[47] Z. Cheng and L. Chen. On security proof of McCullagh-Barreto's key agreement protocol and its variants. To appear in *International Journal of Security and Networks - Special Issue on Cryptography in Networks*. Also available on Cryptology ePrint Archive, Report 2005/201, 2005.

[48] L. Chen and Z. Cheng. Security proof of the Sakai-Kasahara's identity-based encryption scheme. In *Proc. of Cryptography and Coding 2005*, LNCS 3706, pp. 442–459, 2005.

[49] Z. Cheng and R. Comley. Attacks on an ISO/IEC 11770-2 key establishment protocol. *International Journal of Network Security*, Vol.3, No.3, pp. 238–243, 2006.

[50] Z. Cheng, L. Chen, R. Comley and Q. Tang. Identity-based key agreement with unilateral identity privacy using pairings. In *Proc. of ISPEC 2006*, LNCS 3903, pp. 202–213, 2006.

[51] L. Chen, Z. Cheng, J. Malone-Lee and N. Smart. An efficient ID-KEM based on the Sakai–Kasahara key construction. In *IEE Proc. Information Security*, Vol 153(1), pp. 19–26, 2006.

[52] L. Chen, Z. Cheng and N. Smart. Identity-based key agreement protocols from pairings. To appear in *International Journal of Information Security*. Also available on Cryptology ePrint Archive, Report 2006/199, 2006.

[53] Z. Cheng, R. Comley and L. Vasiu. Remove key escrow from the identity-based encryption system. *Foundations of Information Technology in the Era of Network and Mobile Computing*, pp. 37–50, TCS@IFIP, 2004.

[54] R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. In *J. ACM*, 51(4):557–594, 2004.

[55] R. Canetti and J. Herzog. Universally composable symbolic analysis of cryptographic protocols (The case of encryption-based mutual authentication and key exchange). Cryptology ePrint Archive, Report 2004/334, 2004.

[56] R. Canetti, S. Halevi and J. Katz. A forward-secure public-key encryption scheme. In *Proc. of Advances in Cryptology - Eurocrypt 2003*, LNCS 2656, pp. 255–271 2003.

[57] R. Canetti, S. Halevi and J. Katz. Chosen-ciphertext security from identitybased encryption. In *Proc. of Advances in Cryptology - Eurocrypt 2004*, LNCS 3027, pp. 207–222, 2004.

[58] Y. Choie, E. Jeong and E. Lee. Effcient identity-based authenticated key agreement protocol from pairings. *Applied Mathematics and Computation*, 162:179–188, 2005.

[59] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Proc. of Advances in Cryptology - Eurocrypt 2001*, LNCS 2045, pp. 453–474 2001.

[60] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *Proc. of Advances in Cryptology - Eurocrypt 2002*, LNCS 2332, pp. 337–351, 2002

[61] R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *Proc. of Advances in Cryptology - Crypto 2002*, LNCS 2442, pp. 143–161, 2002.

[62] L. Chen and C. Kudla. Identity based authenticated key agreement from pairings. Cryptology ePrint Archive, Report 2002/184, 2002.

[63] L. Chen and C. Kudla. Identity based authenticated key agreement from pairings. In *Proc. of the 16th IEEE Computer Security Foundations Workshop*, pp. 219–233, 2003.

[64] Z. Cheng, M. Nistazakis, R. Comley and L. Vasiu. On the indistinguishability-based security model of key agreement protocols-simple cases. In *Proc. of ACNS 2004*, 2004. Full version avaible on Cryptology ePrint Archive, Report 2005/129.

[65] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33:167–226, 2003.

[66] S. Chatterjee and P. Sarkar. Trading time for space: towards an efficient IBE scheme with short(er) public parameters in the standard model. In *Proc. of ICISC 2005*, LNCS 3935, pp. 424–440, 2005.

[67] Z. Cheng, L. Vasiu and R. Comley. Pairing-based one-round tripartite key agreement protocols. Cryptology ePrint Archive, Report 2004/079, 2004.

[68] A. Dent. A survey of certificateless encryption schemes and security models. Cryptology ePrint Archive, Report 2006/211, 2006.

[69] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22 (6), pp. 644–654, 1976.

[70] A. Dent, B. Libert and K. Paterson, Certificateless encryption schemes strongly secure in the standard model. Cryptology ePrint Archive, Report 2007/121, 2007.

[71] W. Diffie, P. van Oorschot and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2, 107–125 (1992).

[72] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[73] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proc. of Advances in Cryptology - Crypto '99*, LNCS 1666, pp. 535–554.

[74] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. *IEICE Trans. Fund.*, E83-9(1):24–32, 2000.

[75] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA–OAEP is secure under the RSA assumption. In *Proc. of Advances in Cryptology - Crypto '01*, LNCS 2139, pp. 260–274, 2001.

[76] M. Girault. Self-certified public keys. In *Proc. of Advances in Cryptology - Eurocrypt '91*, LNCS 547, pp. 490–497, 1992.

[77] O. Goldreich. Foundations of cryptography – Volume 1. Cambridge University Press, 2001.

[78] O. Goldreich. Foundations of cryptography – Volume 2. Cambridge University Press, 2004.

[79] D. Galindo. Boneh-Franklin identity based encryption revisited. In *Proc. of the 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, LNCS 3580, pp. 791–802, 2005. Also aviable on Cryptology ePrint Archive, Report 2005/117.

[80] P. Gutmann. PKI: it's not dead, just resting. *IEEE Computer*, 35(8):41-49, 2002.

[81] C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Proc. of Advances in Cryptology - Eurocrypt 2003*, LNCS 2656, pp. 272-293, 2003.

[82] C. Gentry. Practical identity-based encryption without random oracles. In *Proc. of Advances in Cryptology - Eurocrypt 2006*,LNCS 4004, pp. 445–464, 2006.

[83] P. Gaudry, F. Hess and N. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *J. Cryptology*, 15(1):19–46, 2002.

[84] S. Galbraith, K. Harrison and D. Soldera. Implementing the Tate pairing. In *Proc. of Algorithmic Number Theory Symposium V*, LNCS 2369, pp. 324–337, 2002.

[85] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, Vol. 28, pp. 270–299, 1984.

[86] D. Galindo, P. Morillo and C. Ràfols. Breaking Yum and Lee generic constructions of certificate-less and certificate-based encryption schemes. In *Public Key Infrastructure: Third European PKI Workshop (EuroPKI 2006)*, LNCS 4043, pp. 81–91, 2006.

[87] R. Granger, D. Page and N. P. Smart. High security pairing-based cryptography revisited. In *Proc. of Algorithmic Number Theory Symposium VII*, LNCS 4076, pp. 480–494, 2006.

[88] S. Galbraith, K. Paterson and N.P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006.

[89] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Proc. of Advances in Cryptology - 2002*, LNCS 2501, pp. 548-566, 2002.

[90] R. Granger and N. P. Smart. On computing products of pairings. Cryptology ePrint Archive, Report 2006/172, 2006.

[91] S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[92] S. Halevi. A plausible approach to computer-aided cryptographic proofs. Cryptology ePrint Archive, Report 2005/181, 2005.

[93] F. Hess. Efficient identity based signature schemes based on pairings. In *Proc. of Selected Areas in Cryptography 9th Annual International Workshop, SAC 2002*, LNCS 2595, pp. 310–324, 2003.

[94] L. Hitt. On an improved definition of embedding degree. Cryptology ePrint Archive, Report 2006/415, 2006.

[95] D. Harkins and D. Carrel. The Internet key exchange protocol (IKE). IETF RFC 2409, Nov. 1998.

[96] G. Horn, K. Martin and C. Mitchell. Authentication protocols for mobile network environment value-added serivices. *IEEE Transactions on Vehicular Technology*, 51(2):383–392, 2002.

[97] D. Hofheinz, J. Müller-Quade and R. Steinwandt. Initiator-resilient universally composable key exchange. In *Proc. of ESORICS 2003*, LNCS 2808, pp. 61–84, 2003.

[98] F. Hess, N.P. Smart and F. Vercauteren. The Eta pairing revisited. Cryptology ePrint Archive, Report 2006/110, 2006.

[99] Q. Huang and D. S. Wong. Generic certificateless encryption in the standard model. Cryptology ePrint Archive, Report 2007/095, 2007.

[100] ISO/IEC 11770-3:1999. Information technology - Security techniques - Key management - Part 3: Mechanisms using asymmetric techniques. 1999.

[101] ISO/IEC 18033-4:2005. Information technology - Security techniques - Encryption algorithms - Part 2: Asymmetric ciphers, 2005.

[102] A. Joux. A one-round protocol for tripartite Diffie-Hellman. In *Proc. of Algorithm Number Theory Symposium – ANTS-IV*, LNCS 1838, pp. 385–394, 2000.

[103] M. Joye, J.-J. Quisquater and M. Yung. On the power of misbehaving adversaries and security analysis of the original EPOC. In *Proc. of CT-RSA 2001*, LNCS 2020, pp. 208–222, 2001.

[104] J. K. Liu, M. H. Au and W. Susilo. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. To appear in *Proc. of ACM AsiaCCS 2007.*

[105] N. Koblitz. Elliptic curve cryptosystems, *Math. Comp.* 48, pp. 203–209, 1987.

[106] H. Krawczyk. HMQV: a high-performance secure Diffie-Hellman protocol. In *Proc. Advances in Cryptology C CRYPTO 2005*, LNCS 3621 pp. 546–566, 2005.

[107] B. Kaliski. Cryptography and security: the narrow road from theory to practice. Invited talk, IPSEC 2006.

[108] E. Kiltz. Chosen-ciphertext secure identity-based encryption in the standard model with short ciphertexts. Cryptology ePrint Archive, Report 2006/122, 2006.

[109] C. Kudla. Special signature schemes and key agreement protocols. PhD Thesis, Royal Holloway University of London, 2006.

[110] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. Cryptology ePrint Archive, Report 2005/076, 2005.

[111] N. Koblitz and A. Menezes. Another look at "provable security" II. Cryptology ePrint Archive, Report 2006/229, 2006.

[112] B. G. Kang and J. H. Park. Is it possible to have CBE from CL-PKE?. Cryptology ePrint Archive, Report 2005/431, 2005.

[113] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proc. of 10th ACM Conf. Computer and Communications Security*, pp. 155–164, 2003.

[114] T. Kitagawa, P. Yang, G. Hanaoka, R. Zhang, H. Watanabe, K. Matsuura and H. Imai. Generic transforms to acquire CCA-Security for identity based encryption: the cases of FOpkc and REACT. In *Proc. of ACISP 2006*, LNCS 4058, pp. 348–359, 2006.

[115] C. H. Lim, P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Proc. of Advances in Cryptology - Crypto '97*, LNCS 1294, pp. 249–263, 1997.

[116] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, **28**, 119–134, 2003.

[117] B. Libert and J.-J. Quisquater. Identity based encryption without redundancy, In *Proc. of the 3rd Applied Cryptography and Network Security conference (ACNS'05)*, LNCS 3531, pp. 285–300, 2005

[118] B. Libert and J.-J. Quisquater. On constructing certificateless cryptosystems from identity based encryption. In *Proc. of Public Key Cryptography 2006 (PKC'06)*, LNCS 3958, pp. 474–490, 2006

[119] S. Li, Q. Yuan and J. Li. Towards secure two-part authenticated key agreement protocols. Cryptology ePrint Archive, Report 2005/300, 2005.

[120] V. Miller. Use of elliptic curves in cryptography. In *Proc. Advances in cryptology - Crypto '85*, LNCS 218, pp. 417–426, 1986.

[121] V. Miller. The Weil pairing and its efficient calculation. *J. Cryptology*, 17(4):235-261. 2004.

[122] J. Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS#1 v2.0. In *Proc. of Advances in cryptology - Crypto 2001*, LNCS 2139, pp. 230–238, 2001.

[123] A. Menezes. Another look at HMQV. To appear in *Journal of Mathematical Cryptology*. Available at Cryptology ePrint Archive, Report 2005/205, 2005.

[124] N. McCullagh and P. Barreto. A new two-party identity-based authenticated key agreement. Cryptology ePrint Archive, Report 2004/122, 2004.

[125] N. McCullagh and P. Barreto. A new two-party identity-based authenticated key agreement. In *Proc. of CT-RSA 2005*, LNCS 3376, pp. 262–274, 2005.

[126] A. J. Menezes, T. Okamoto and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Info. Theory*, IT-39, pp. 1639–1646, 1993.

[127] A. J. Menezes, P. C. Van Oorschot and S. A. Vanstone. Handbook of applied cryptography. CRC Press, 1997.

[128] S. Mitsunari, R. Sakai and M. Kasahara. A new traitor tracing. *IEICE Trans.* Vol. E85-A, No.2, pp. 481–484, 2002.

[129] A. Menezes and B. Ustaoglu. On the importance of public-key validation in the MQV and HMQV key agreement protocols. In *Proc. of Progress in Cryptology - Indocrypt 2006*, LNCS 4329, pp. 133–147, 2006.

[130] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case. In *Proc. Advance in Cryptology-Crypto 2002*, LNCS 2442, pp. 111–126, 2002.

[131] D. Naccache. Secure and practical identity-based encryption. Cryptology ePrint Archive, Report 2005/369, 2005.

[132] National Institute of Standards and Technology. NIST recommendation for key management part 1: general. NIST Special Publication 800-57. August, 2005.

[133] E. Okamoto. Proposal for identity-based key distribution system. *Electronics Letters*, 22:1283–1284, 1986.

[134] T. Okamoto and D. Pointcheval. REACT: rapid enhanced-security asymmetric cryptosystem transform. In *Proc. of Topics in Cryptology CT-RSA 2001*, LNCS 2020, pp. 159C-174, 2001.

[135] O. Pereira. Modelling and security analysis of authenticated group key agreement protocols. Dissertation, 2003

[136] K. G. Paterson and J. C. N. Schuldt. Efficient identity-based signatures secure in the standard model. In *Proc. of ACISP 2006*, LNCS 4058, pp. 207–222, 2006.

[137] M. Scott. Authenticated ID-based key exchange and remote log-in with insecure token and PIN number. Cryptology ePrint Archive, Report 2002/164, 2002.

[138] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of Advances in Cryptology - Crypto '84*, LNCS 196, pp. 47–53, 1984.

[139] H. Shacham. New paradigms in signature schemes. PhD Thesis, U. Stanford, 2005.

[140] J. H. Silverman. The arithmetic of elliptic curves. Graduate Texts in Mathematics, Vol. 106, Springer-Verlag, 1986.

[141] I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curves in characteristic p. *Math. Comp.*, 67, pp. 353–356, 1998.

[142] N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, Vol 12(3), pp. 193–196, 1999.

[143] N. P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters* 38, pp. 630–632, 2002.

[144] K. Shim. Efficient ID-based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 39(8), pp. 653–654, 2003.

[145] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. of Advances in Cryptology - Eurocrypt '97*, LNCS 1233, pp. 256–266, 1997.

[146] V. Shoup. On formal models for secure key exchange. Theory of Cryptography Library, 1999.

[147] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004.

[148] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Commentarii Math.* Univ. St. Pauli, 47, pp. 81–92, 1998.

[149] H. Sun and B. Hsieh. Security analysis of Shim's authenticated key agreement protocols from pairings. Cryptology ePrint Archive, Report 2003/113, 2003.

[150] Y. Shi, J. Li, J. Pan and J. Shi. Efficient certificateless public key encryption with pairing. In *Proc. of Networks and Communication Systems 2006*, 2006.

[151] R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003.

[152] R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing. In *Proc. of 2000 Symposium on Cryptography and Information Security*, Japan, 2000.

[153] R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing over elliptic curve (in Japanese). In *Proc. of The 2001 Symposium on Cryptography and Information Security*, Oiso, Japan, January 2001.

[154] V. Shoup and A. Rubin. Session key distribution using smart cards. In *Proc. of Advances in Cryptology - Eurocrypt '96*, LNCS 1070, 1996.

[155] T. Saito and S. Uchiyama. A remark on the MOV algorithm for non-supersingular elliptic curves. *IEICE Trans. Fundamentals*, Vol. E84-A, No. 5, pp. 1266–1268, 2001.

[156] N. P. Smart and F. Vercauteren. On computable isomorphisms in efficient pairing based systems. Cryptology ePrint Archive, Report 2005/116, 2005.

[157] O. Schirokauer, D. Weber and T. Denny. Discrete logarithms: the effectiveness of the index calculus method. In *Proc. of ANTS II*, LNCS 1122, pp. 337–351, 1996.

[158] K. Tanaka and E. Okamoto. Key distribution system for mail systems using ID-related information directory. *Computers & Security*, 10:25–33, 1991.

[159] E. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In *Proc. of Advances in Cryptology - Eurocrypt 2001*, LNCS 2045, pp. 195–210, 2001.

[160] Votlage Security. `http://www.voltage.com/`.

[161] Y. Wang. Efficient identity-based and authenticated key agreement protocol. Cryptology ePrint Archive, Report 2005/108, 2005.

[162] B. R. Waters. Efficient identity-based encryption without random oracles. In *Proc. of Advances in Cryptology - Eurocrypt 2005*, LNCS 3494, 114–127, 2005.

[163] G. Xie. An ID-based key agreement scheme from pairing. Cryptology ePrint Archive, Report 2005/093, 2005.

[164] D. H. Yum and P. J. Lee. Generic construction of certificateless encryption. In *Proc. of Computational Science and Its Applications ICCSA 2004: Part I*, LNCS 3043, pp. 802–811, 2004.

[165] D. H. Yum and P. J. Lee. Identity-based cryptography in public key management. In *Proc. of Public Key Infrastructure: First European PKI Workshop (EuroPKI 2004)*, LNCS 3093, pp. 71–84, 2004.

[166] Q. Yuan and S. Li. A new effcient ID-based authenticated key agreement protocol. Cryptology ePrint Archive, Report 2005/309, 2005.

[167] P. Yang, T. Kitagawa, G. Hanaoka and *et al.* Applying Fujisaki-Okamoto to identity-based encryption. In *Proc. of AAECC 2006*, LNCS 3857, pp. 183–192, 2006.

[168] Z. Zhang and D. Feng. On the security of a certificateless public-key encryption. Cryptology ePrint Archive, Report 2005/426, 2005.

[169] F. Zhang, R. Safavi-Naini and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In *Proc. of International Workshop on Practice and Theory in Public Key Cryptography - PKC 2004*, LNCS 2947, pp. 277–290, 2004